BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam pelaksanaan praktik kerja magang ini, mahasiswa diberikan kesempatan sebagai posisi *Programmer* dalam departemen *Web & Android*. Peran mahasiswa pada divisi ini adalah membantu pekerjaan dan/atau projek yang berkaitan dengan pengembangan aplikasi perusahaan dari sisi *back-end*. Adapun anggota dari departemen yang bertanggung jawab secara langsung untuk mengawasi mahasiswa adalah Bapak Yos Sugianto sebagai *User*, Bapak Febrinanda Endriz Pratama sebagai *Supervisor*, dan Bapak Fery Kurniawan sebagai *Senior Front-end Developer*. Komunikasi antar anggota untuk keperluan pekerjaan dilakukan melalui aplikasi Slack, dan kolaborasi projek melalui local gitlab perusahaan. Untuk pelaksanaan meeting dan diskusi dilakukan secara langsung di ruang meeting.

3.2 Tugas yang Dilakukan

Pelaksanaan tugas kerja magang yang dilakukan mahasiswa biasa diberikan oleh User via *general meeting*. Adapun cakupan dari tugas yang dilakukan dapat berupa end-user testing sampai pengembangan aplikasi. Untuk rincian tugas-tugas yang dilakukan akan dijelaskan pada poin-poin berikut:

1. ***Spring Boot***

   Projek ini meliputi penggunaan bahasa pemrograman *Java* dengan bantuan framework Spring Boot. Spring Boot merupakan kerangka kerja berbasis Java yang dirancang untuk mensimplifikasi pengembangan aplikasi berbasis Spring. Penggunaan Spring Boot memungkinkan pengguna untuk mengelola dependensi secara otomatis melalui Spring Initializr, akses ke *embedded server* seperti Tomcat, dan memiliki struktur projek yang terorganisir. Fokus utama dalam pengembangan aplikasi Spring adalah untuk membuat *RESTful API*, sebuah protokol komunikasi *cacheable* yang menyambungkan *client* dengan *server* via *HTTP method*. Dalam praktik kerja magang ini, mahasiswa melakukan pengembangan aplikasi *ticketing* dan *event management system* (EMS) menggunakan *Spring Boot*. Adapun langkah pengembangan aplikasi

yang dilakukan mahasiswa adalah sebagai berikut:

(a) Membuat kerangka awal dan *dependencies* yang dibutuhkan dengan Spring Initializr, kemudian meng-ekstrak projek ke dalam workspace seperti Intellij Idea.

(b) Mengkonfigurasi koneksi database dan pengaturan lain di application.properties.

(c) Pengembangan komponen utama aplikasi (Model, Repository, Service, dan Controller).

(d) Mengakses dan melakukan endpoint testing melalui Postman untuk memastikan aplikasi berjalan dengan lancar.

(e) Mendokumentasikan seluruh endpoint, termasuk cara melakukan request dan hasil yang diperoleh ke dalam *api documentation*.

2. ***HRIS***

Human Resource Information System atau disingkat HRIS adalah salah satu aplikasi internal PT Panca Budi Pratama dengan fungsi utama untuk mengelola data karyawan. Sistem ini menyimpan informasi personal, data kehadiran, flow perizinan & persetujuan, jatah cuti, dan lainnya. Salah satu fitur utama dalam aplikasi, yaitu sistem perizinan sedang dalam pengembangan dan memerlukan *user testing*. Maka dengan itu, mahasiswa dilibatkan dalam *end-user testing* untuk memastikan fitur bekerja sesuai dengan ekspektasi. Langkah pengerjaan testing adalah sebagai berikut:

(a) Menentukan *workflow* perizinan yang akan dilakukan testing.

(b) Mencatat ekspektasi dari *workflow* perizinan menggunakan Notion, dan kemudian melakukan testing workflow tersebut via aplikasi.

(c) Mencatat hasil aktual yang diperoleh ke dalam Notion dan memberikan komen apabila tidak sesuai ekspektasi.

(d) Mengadakan rapat mingguan dengan tim pengembang untuk menyampaikan *feedback* dan hasil testing.

3. ***Odoo***

Odoo merupakan ERP (Enterprise Resource Planning) open-source yang digunakan oleh perusahaan untuk mengelola proses bisnis secara otomatis.

Peran mahasiswa disini adalah mendukung pengembangan custom module menggunakan bahasa pemrograman Python yang dibantu dengan basis data PostgreSQL. Pembuatan dan debugging module dilakukan dengan menggunakan Pycharm sebagai integrated development environment (IDE). Mahasiswa diberikan kesempatan untuk melakukan dua projek berskala kecil dengan Odoo, yaitu mengembangkan module: *sales credit limit* untuk mengatur batas pinjaman yang diperbolehkan untuk setiap *customer*; dan *sales target* yang berfungsi untuk menghitung penjualan daring (*Invoicing*) dan luring (*Point of Sale*). Secara umum, pengerjaan projek di Odoo memiliki langkah-langkah seperti:

(a) Membuat struktur awal module sesuai dengan standar Odoo, yang terdiri dari folder seperti `models`, `views`, dan `security`.

(b) Mendefinisikan model data (class Python) dengan menggunakan inheritance untuk merepresentasikan struktur logika bisnis yang diinginkan.

(c) Membuat tampilan antarmuka pengguna (UI) dalam bentuk XML view seperti form, tree (list), dan kanban view.

(d) Melakukan instalasi dan testing module melalui antarmuka Odoo untuk memastikan bahwa fungsi yang dikembangkan berjalan sesuai dengan kebutuhan.

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang di PT Panca Budi Pratama (PBP) untuk setiap minggu diuraikan seperti pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Memahami konsep dasar Spring Boot dan membuat projek CRUD dasar sebagai latihan, kemudian melakukan testing melalui Postman.
2	Diberikan projek setengah jadi mengenai sistem ticketing/complaint, mahasiswa melengkapi program (Master Entities dan Transactions) dan melakukan unit test.
3	Melakukan endpoint testing via Postman dan membuat API documentation untuk projek ticketing.
4	Rapat dengan divisi untuk membahas projek event dan kemudian memulai pengerjaan.
5	Pembuatan Master Entities (Kategori & Lokasi) dan Transaksi Create Event.
6	Penyelesaian projek event dengan membuat Transaksi (Daftar event, Reschedule, dan Pendaftaran Karyawan). Melakukan testing via Postman dan membuat API documentation.
7	Melakukan end-user testing untuk sistem perizinan di aplikasi HRIS. Mencatat ekspektasi, hasil, dan bug melalui Notion.
8	Melanjutkan end-user testing dan mencatat ekspektasi, hasil, dan bug melalui Notion.
9	Melanjutkan end-user testing dan mempelajari konsep dasar ERP Odoo.
10	Melakukan instalasi Odoo, PyCharm, dan PostgreSQL. Belajar grammar dan flow pembuatan module Odoo via PyCharm.
11	Melakukan praktik Odoo dengan membuat custom module dengan berbagai jenis field (Float, Selection, Boolean, dll) serta function call.
12	Menambahkan git oca credit control ke dalam Odoo dan menambahkan module credit limit customer.
13	Membuat projek Odoo sales target untuk menghitung jumlah penjualan daring dan luring. Memulai pembuatan model & views.
14	Mengembangkan fitur date filter dan tabel invoice target history

3.3.1 Minggu ke-1: Perkenalan Tempat Kerja dan Projek Kecil Spring Boot

Pada minggu pertama praktik kerja magang, mahasiswa dipertemukan dengan seluruh anggota dari divisi IT dan dijelaskan mengenai tugas yang akan dijalani mahasiswa selama durasi magang. Mahasiswa diperkenalkan dengan *Supervisor* Bapak Febrinanda Endriz Pratama sebagai pembimbing dan pemandu dan juga *User* Bapak Yos Sugianto. Mahasiswa ditempatkan dalam posisi Programmer bersama divisi *Back-end* dan ditugaskan untuk bekerja sama serta membantu projek-projek yang sedang dikerjakan oleh perusahaan dari sisi *back-end*. Secara spesifik, mahasiswa memiliki tugas untuk mengembangkan logika aplikasi, mengelola basis data, dan memastikan aplikasi dapat berjalan lancar sesuai ekspektasi.

Untuk penugasan pertama, mahasiswa diarahkan untuk mempelajari dasar-dasar pengembangan aplikasi menggunakan *framework* Spring Boot. Pembelajaran dilakukan secara mandiri dan dibimbing oleh *supervisor* melalui diskusi via Slack. Sebagai latihan awal, mahasiswa diarahkan untuk membuat sebuah program sederhana bernama api-sekolah, yang berfungsi untuk menyimpan data siswa dan kelas yang diambil.

A Infrastruktur Program api-sekolah

Pengembangan pada projek api-sekolah menggunakan bahasa pemrograman *Java* dengan *framework* Spring Boot. Pengembangan dibantu dengan *dependencies* utama Spring Boot Starter Web untuk RESTful API, serta Spring Data JPA untuk integrasi dengan basis data. Database yang digunakan adalah MySQL, dengan konfigurasi koneksi diletakkan dalam file application.properties; dan Database Management System (DBMS) yang digunakan untuk mengelola basis data adalah MySQL Workbench.

Gambar 3.1. Logo Spring Boot

B Alur Pengembangan Program api-sekolah

Alur pengembangan program ini mengacu pada arsitektur MVC (Model-View-Controller) Spring Boot. Dalam Spring, konsep View pada umumnya berupa format JSON sebagai response dari Controller. Adapun lapisan arsitektur MVC dalam projek Spring ini adalah sebagai berikut:

1. Model: Merepresentasikan data atau entitas yang akan disimpan dalam basis data. Sebagai contoh, entitas Siswa dapat berisi atribut seperti `id`, `nama`, `kelas`, dan `umur`. Dalam Spring, Model perlu diberikan anotasi JPA seperti `@Entity` dan `@ID` agar dapat dikonversikan ke dalam tabel basis data.

2. Repository: Merupakan interface yang memperluas `JpaRepository` dan berfungsi sebagai'medium komunikasi kepada basis data. Repository menyediakan operasi *query* dasar seperti `@findAll()`, `@findById`, `@save`, dan `@deleteById` yang ditulis secara langsung dalam aplikasi Spring.

3. Service: Memiliki peran sebagai perantara antara Controller dan Repository; dan tempat logika utama aplikasi berada. Misal, menambahkan siswa ke dalam kelas tertentu, mencari siswa dalam umur tertentu, dan sebagainya. Service perlu ditandai dengan anotasi @Service.

4. Controller: Merupakan lapisan yang menangani request dari client (biasanya HTTP request). Controller menggunakan anotasi seperti `@RestController`, `@RequestMapping`, `@PostMapping`, `@GetMapping`, dan lainnya untuk memetakan endpoint. Di dalam controller, request akan diteruskan ke service untuk diproses dan kemudian response dikembalikan ke client.

C Pengerjaan Program api-sekolah

Pengerjaan dimulai dengan pembuatan endpoint dasar (Create dan Read) untuk masing-masing *Entity*, yaitu `Classroom` dan `Student`. Sebuah kelas dibuat untuk menampung banyak siswa, sehingga Student memiliki relasi many-to-one terhadap Classroom. Pada tahapan awal ini, kedua Entity memiliki kelas Service dan Controller masing-masing. Namun, banyaknya metode yang memiliki struktur dan fungsi sama pada kedua Entity membuat program kurang efisien. Maka

program di revisi dengan menggunakan konsep Generic untuk meringkas dan menghindari adanya pengulangan kode. Pengembangan program ini diakhiri dengan pengujian via Postman.

D Penggunaan API

API yang dibuat dalam projek ini mencakup: menunjukkan semua data, menunjukkan data dengan ID tertentu, dan menyimpan sebuah data. Data yang dimaksud di sini adalah data dari entitas `Student` dan `Classroom`. Melanjutkan penjelasan sebelumnya, konsep *Generic* menghasilkan dua lapis arsitektur dalam penggunaan API. *StudentController* dan *ClassroomController* masing-masing memiliki RequestMapping untuk dirinya sendiri (`.../students atau .../classrooms`), dan GenericController menambahkan endpoint untuk menghasilkan 3 API yang disebut di atas.

```
public abstract class GenericController<T, ID> {  2 usages  2 inheritors  GilbertOng
    protected final GenericService<T, ID> service;  4 usages

    public GenericController(GenericService<T, ID> service) { this.service = service; }

    @GetMapping("all")  no usages  GilbertOng
    public List<T> getAll() { return service.findAll(); }

    @GetMapping("/{id}")  no usages  GilbertOng
    public Optional<T> getById(@PathVariable ID id) { return service.findById(id); }

    @PostMapping("/add")  no usages  GilbertOng
    public T addEntity(@RequestBody T entity) { return service.save(entity); }
```

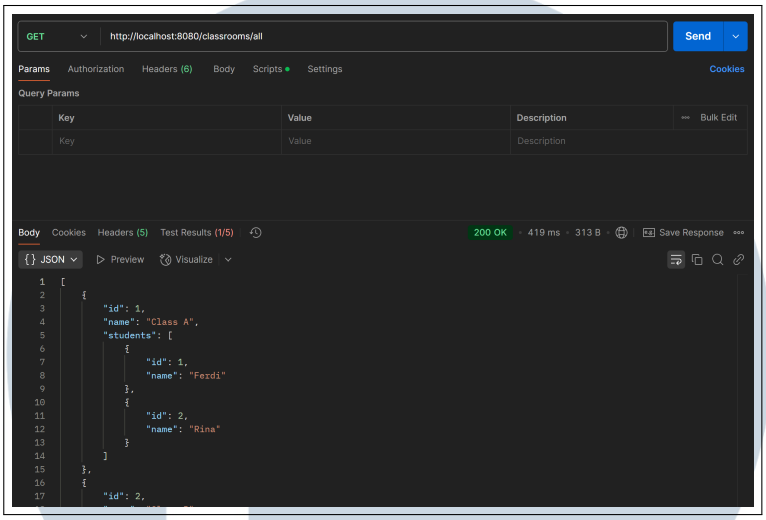
Gambar 3.2. Generic Controller Program api-sekolah

Struktur `Generic Controller` yang ditunjukkan pada gambar 3.2 memiliki tiga endpoint utama yang berfungsi untuk melanjutkan turunan kedua controller. Masing-masing endpoint tersebut bertanggung jawab terhadap operasi dasar dalam sebuah sistem RESTful:

1. `@GetMapping("all")` → Menampilkan Semua Data

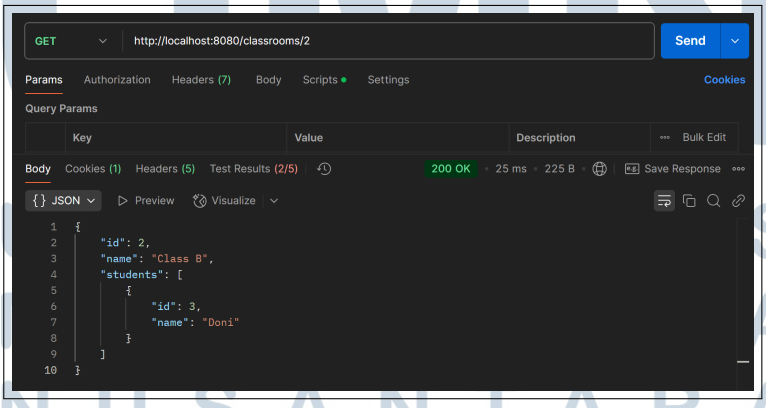
   Endpoint ini bertugas untuk mengembalikan semua data dari entitas yang dimaksud. Misalnya, jika dipanggil melalui route `/student/all`, maka akan dikembalikan seluruh data `Student` yang tersimpan di database; dan sebaliknya, begitu pula dengan route `/classrooms/all`. Hal ini dikarenakan oleh method `findAll()` dari service yang telah di-*inject* melalui konstruktor

Generic pula. Contoh hasil dari penggunaan API *get classrooms* dapat dilihat di gambar 3.3



Gambar 3.3. Hasil Percobaan API *GET Classrooms*

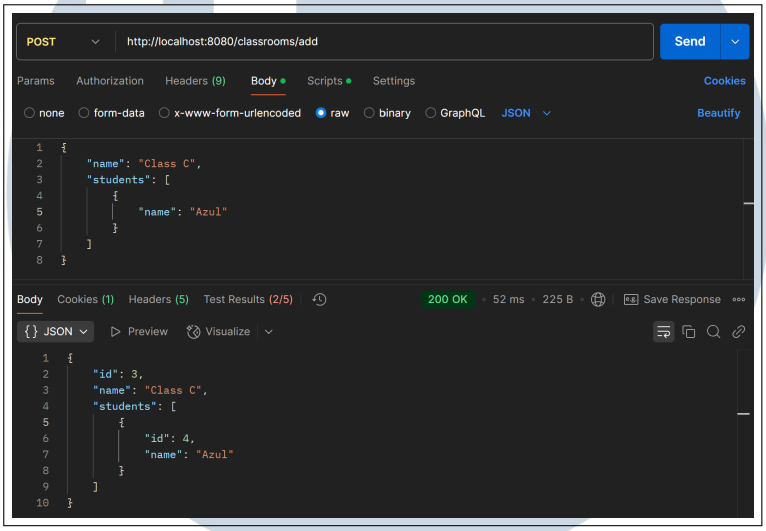
2. `@GetMapping("/{id}")` → Menampilkan Satu Data Berdasarkan ID Endpoint ini akan mengembalikan data spesifik berdasarkan ID entitas. Nilai ID diambil dari path variable pada URL dan diteruskan ke method `findById()` di service. Sebagai contoh, permintaan `GET /classroom/5` akan mengembalikan detail dari `Classroom` dengan ID 5, jika ada; dan permintaan `GET /students/5` akan mengembalikan detail dari `Students` dengan ID 5. Penggunaan API get classrooms by id seperti di gambar 3.4



Gambar 3.4. Hasil Percobaan API *GET Classrooms by ID*

3. `@PostMapping("/add")` → Menyimpan Data Baru Endpoint ini menerima permintaan `POST` dengan format JSON dalam *request body* yang mewakili

data entitas yang ingin disimpan. JSON tersebut di-*bind* ke objek generik `T`, kemudian diteruskan ke method `save()` dari service. Misalnya, permintaan `POST /classrooms/add` dengan JSON yang berisi nama kelas dan entitas *Students* akan menghasilkan data baru di tabel *Classrooms*. Penggunaan API add classrooms dapat dilihat pada gambar 3.5



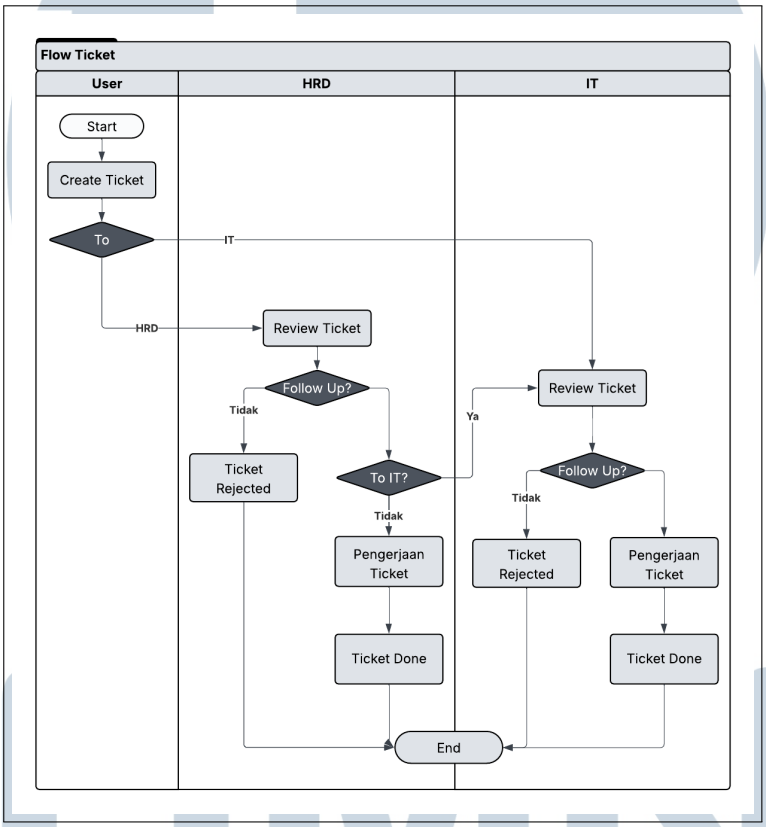
Gambar 3.5. Hasil Percobaan API *Add Classrooms*

3.3.2 Minggu ke-2 dan ke-3: Projek Ticketing/Complaint System

Pengembangan projek Ticketing masih menggunakan infrastruktur yang sama dengan sebelumnya, yaitu Java Spring Boot dan MySQL. Projek ini sendiri ditujukan untuk memudahkan karyawan internal dalam membuat, mengelola, dan menindaklanjuti laporan keluhan atau permintaan bantuan secara terstruktur. Sistem ini diharapkan dapat membantu manajemen dalam memonitor status tiket yang masuk, mempercepat proses penyelesaian, serta menyediakan dokumentasi yang rapi terkait semua aktivitas balasan/*follow-up* yang dilakukan. Mahasiswa ditempatkan dalam tim pengembangan back-end, dengan tanggung jawab utama mencakup perancangan struktur entitas, pembuatan logika bisnis melalui service layer, serta pengujian endpoint untuk semua operasi CRUD menggunakan Postman.

Alur sistem ticketing sendiri dimulai dari pembuatan tiket oleh user, di mana tiket dapat ditujukan kepada divisi HRD atau IT. Setelah tiket dibuat, divisi yang dituju akan melakukan proses *review ticket* untuk menilai validitas dan kelengkapan informasi. Dalam tahap ini, tiket dapat langsung ditolak apabila dianggap tidak layak atau tidak relevan. Jika diperlukan, tiket juga dapat diteruskan ke divisi lain

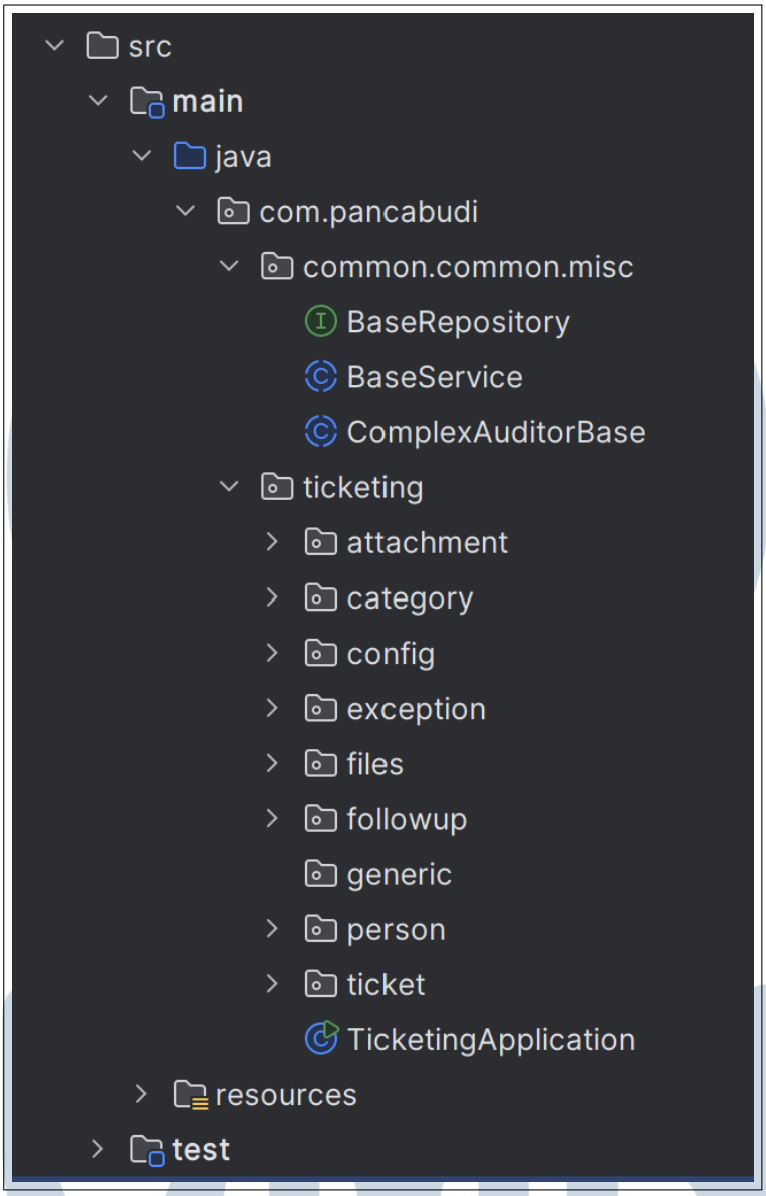
untuk ditindaklanjuti. Apabila tiket tetap ditangani oleh divisi yang sama, maka akan masuk ke tahap *pengerjaan tiket*, di mana proses penyelesaian dilakukan oleh staf terkait. Setelah pengerjaan selesai, tiket akan diberi status `Done` sebagai tanda bahwa masalah telah diselesaikan. Alur ini dirancang untuk memastikan bahwa setiap tiket diproses secara sistematis, transparan, dan sesuai tanggung jawab divisi yang berwenang. Untuk visualisasi alur dapat dilihat lebih jelas pada gambar 3.6



Gambar 3.6. Flow Pembuatan Ticket

A Pembuatan Master Entities

Master Entity adalah sebuah istilah yang merujuk pada entitas yang dapat berdiri sendiri, memiliki data utama, dan sering menjadi rujukan oleh entitas lain melalui relasi (*e.g. foreign key*). Dalam projek ticketing ini, proses pengembangan diawali dengan pembuatan dua *Master Entity* utama, yaitu Master Ticket Category dan Master Follow-up. Kedua entitas ini menjadi fondasi bagi transaksi tiket karena setiap tiket yang dibuat wajib mengacu pada kedua entitas tersebut. Adapun struktur projek seperti di gambar 3.7



Gambar 3.7. Struktur Aplikasi Ticketing

A.1 Master Ticket Category

Entitas ini menyimpan informasi kategori dari setiap tiket yang masuk, seperti ”Permintaan IT”, ”Keluhan Fasilitas”, atau ”Administrasi Umum”. Setiap pembuatan tiket harus disambungkan dan memiliki referensi kategori. Entitas ini memiliki atribut utama seperti *id*, *name*, dan *timestamp (created date & updated date)*, serta dibuat endpoint CRUD standar untuk memungkinkan admin sistem menambah, membaca, memperbarui, atau menghapus kategori yang tersedia. Untuk kasus menghapus kategori, tidak dilakukan *hard delete*, melainkan

menggunakan variabel *boolean disable* agar tidak ditampilkan dalam penggunaan API. Validasi tambahan juga ditambahkan untuk memastikan nama kategori bersifat unik dan tidak kosong.

A.2 Master Follow-up

Master Follow-Up merupakan entitas utama yang berperan penting dalam mendefinisikan tindakan lanjutan (*follow-up*) yang tersedia setelah sebuah tiket ditinjau oleh divisi terkait. Entitas ini digunakan sebagai acuan dalam proses pengambilan keputusan terhadap tindak lanjut tiket, apakah akan diteruskan ke divisi lain atau langsung ditangani oleh divisi yang bersangkutan. Adapun atribut yang dimiliki entitas ini sama seperti *ticket category*, yauti *id*, *name*, dan *timestamp (created date & updated date)*. Dalam projek ini, Follow-up yang diperlukan hanya berupa HRD dan IT. Dengan pendekatan ini, ticket-ticket dapat lebih mudah dikelola dan disortir berdasarkan divisi yang ditujukan.

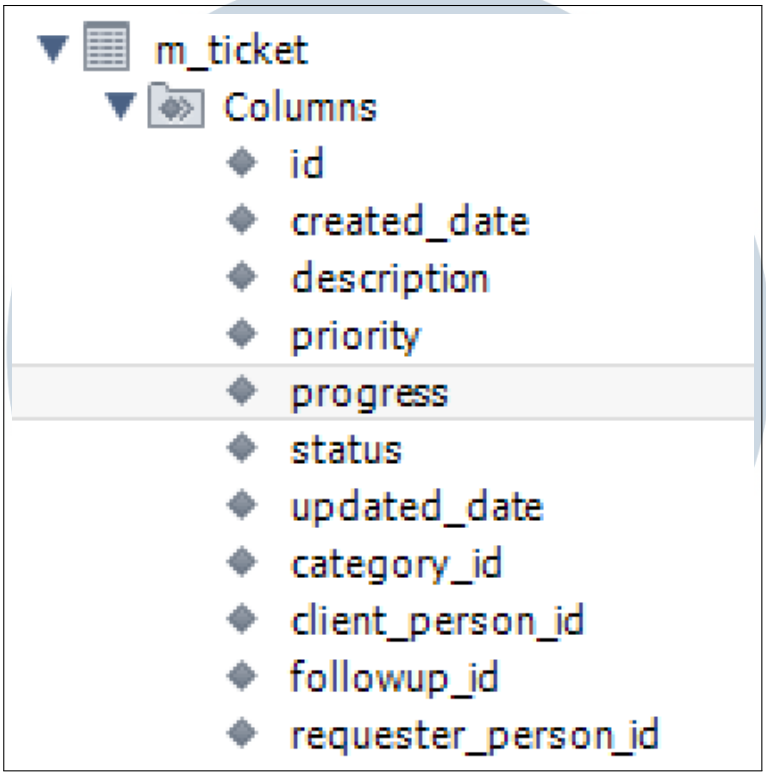
B Pembuatan Transactions

Setelah pembuatan master entities selesai, tahap selanjutnya adalah mengembangkan transaksi utama yang menjadi inti dari sistem Ticketing. Transaksi ini mencakup proses pembuatan tiket, pemberian tanggapan atau tindak lanjut, hingga pencatatan aktivitas yang terjadi selama proses penyelesaian. Transaksi-transaksi ini didesain dengan mengikuti pola relasional, di mana masing-masing entitas saling terhubung dan memiliki foreign key ke master entity yang relevan. Untuk menjaga konsistensi kode dan efisiensi pengembangan, pendekatan generic yang sebelumnya telah digunakan juga diterapkan dalam proses pembuatan transaksi ini.

B.1 Ticket

Ticket merupakan entitas utama dalam sistem ini dan menjadi pusat dari seluruh proses bisnis yang dibangun. Setiap tiket merepresentasikan satu laporan, permintaan, atau keluhan yang diajukan oleh pengguna sistem, dan akan ditindaklanjuti oleh divisi yang dituju, seperti HRD atau IT. Entitas ini memiliki relasi langsung ke berbagai *master entity* seperti *Follow-up* dan *Ticket Category*, serta ke entitas transaksi lain seperti *Person*, *Collaborators*, *Attachments*, dan *Logs*. Informasi utama yang disimpan pada entitas ini meliputi nama tiket, deskripsi,

status, prioritas, dan tanggal pembuatan. Struktur tabel *Ticket* dapat dilihat pada gambar 3.8



Gambar 3.8. Struktur Tabel Ticket

B.2 Person

Transaksi *Person* berperan sebagai representasi dari pengguna sistem yang terlibat dalam suatu tiket, baik sebagai pelapor (*requester*), reviewer (*client*), maupun kolaborator (*collaborators*). Entitas ini digunakan dalam konteks transaksi untuk mencatat peran dan keterlibatan seseorang pada setiap aktivitas terkait tiket. Sebagai contoh, ketika sebuah tiket dibuat, informasi siapa yang membuat tiket akan dicatat melalui relasi ke entitas *Person*, begitu pula dengan siapa yang meninjau, menangani, atau memberikan tanggapan terhadap tiket tersebut.

B.3 Collaborators

Entitas *Collaborators* digunakan untuk mendukung kasus di mana suatu tiket memerlukan kontribusi atau keterlibatan lebih dari satu orang dalam proses pengerjaannya. Misalnya, ketika sebuah masalah yang dilaporkan membutuhkan

bantuan dari beberapa individu atau departemen, nama-nama tersebut akan dicatat dalam tabel *Collaborators*. Entitas ini menjembatani relasi antara tiket dan satu atau lebih *Person* sebagai kolaborator, sehingga memungkinkan sistem untuk melacak siapa saja yang turut berkontribusi dalam penyelesaian suatu tiket.

B.4 Attachment

Entitas *Attachment* menyimpan file atau dokumen pendukung yang diunggah oleh pengguna selama proses pembuatan tiket ataupun saat melakukan tindak lanjut. Lampiran ini dapat berupa gambar, dokumen PDF, atau jenis file lainnya yang relevan dengan isi tiket. Setiap entri *Attachment* memiliki relasi langsung ke entitas *Ticket*, sehingga sistem dapat dengan mudah menampilkan atau mengelola file yang terkait. Untuk efisiensi dan keamanan, file biasanya disimpan di direktori server, sementara metadata-nya (seperti nama file, tipe, dan path) disimpan dalam tabel ini.

B.5 Logs

Entitas *Logs* berfungsi sebagai pencatat semua aktivitas yang terjadi dalam siklus hidup sebuah tiket. Setiap perubahan status, perpindahan divisi, tindakan penolakan, hingga penyelesaian tiket akan dicatat dalam entitas ini. Dengan adanya *Logs*, sistem dapat memberikan histori lengkap terhadap sebuah tiket, yang bermanfaat untuk kebutuhan audit, evaluasi performa, maupun analisis kasus berulang. Setiap log mencakup informasi seperti jenis aksi (e.g., created, updated, rejected, reassigned, done), timestamp, pelaku aksi (relasi ke *Person*), dan referensi ke tiket yang terlibat. Penulisan *Logs* ke dalam basis data sudah ter-otomatisasi dan tergantung pada jenis aksi yang dilakukan terhadap tiket tersebut.

C Penggunaan API dalam Aplikasi Ticketing

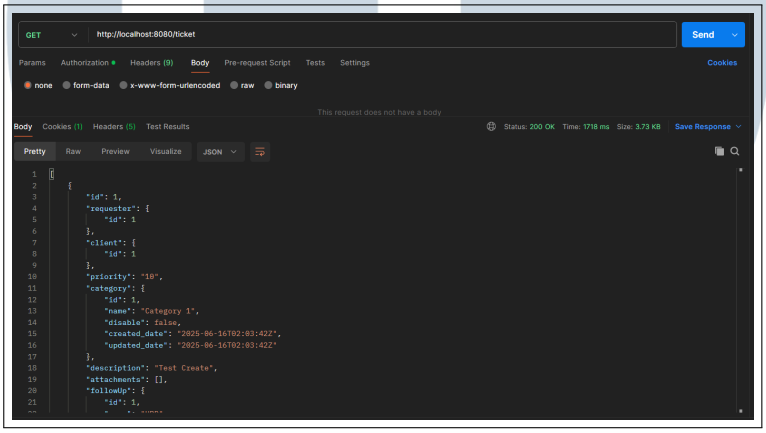
C.1 API Ticket

API Ticket merupakan inti dari sistem karena menangani seluruh proses pembuatan, pembaruan, penghapusan, dan pengambilan data tiket. Endpoint-endpoint yang tersedia mendukung pengelolaan lengkap siklus hidup tiket, mulai dari awal pelaporan hingga penyelesaian. Seluruh endpoint mengikuti standar

RESTful dan dapat diakses dengan method HTTP yang sesuai, seperti GET, POST, PATCH, dan DELETE.

C.1.1 View Semua Ticket

Endpoint ini digunakan untuk menampilkan seluruh data tiket yang tersimpan dalam sistem. Setiap entri akan memuat informasi lengkap, termasuk pelapor, kategori, status, progress, log aktivitas, serta daftar kolaborator yang terlibat. Penggunaan endpoint seperti gambar 3.9



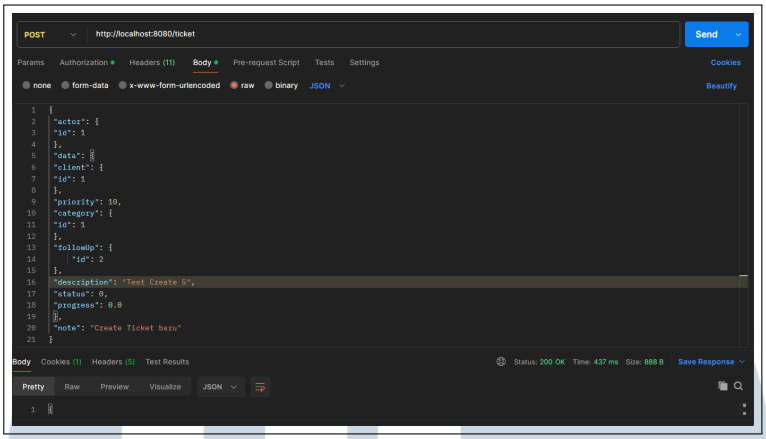
Gambar 3.9. API *View All Ticket*

C.1.2 View Ticket berdasarkan ID

Endpoint ini digunakan untuk mengambil informasi detail dari satu tiket berdasarkan ID-nya. Respons berisi informasi lengkap tiket.

C.1.3 Create Ticket

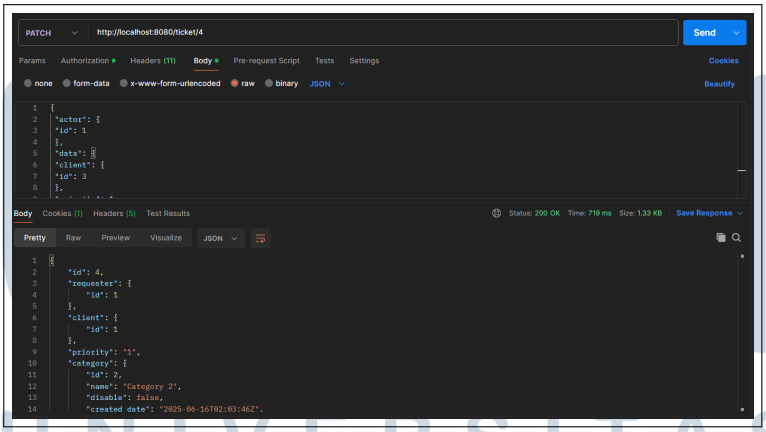
Digunakan untuk membuat tiket baru. Data yang dibutuhkan meliputi ID client, kategori tiket, deskripsi masalah, dan prioritas. Selain itu, actor yang membuat tiket juga perlu dicantumkan dalam struktur JSON. Setelah berhasil, sistem akan memberikan respons berupa objek tiket lengkap beserta log awal. Penggunaan endpoint seperti gambar 3.10



Gambar 3.10. API *Create Ticket*

C.1.4 Update Ticket

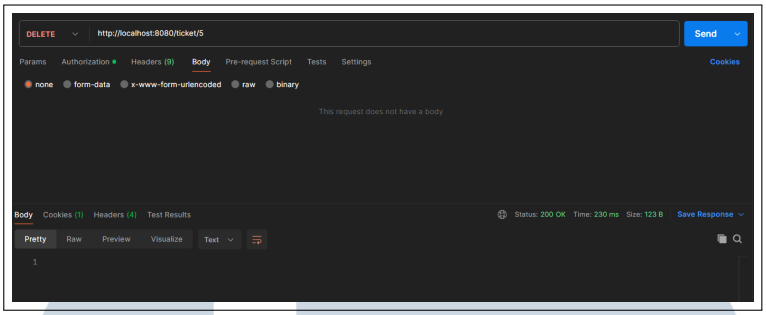
Digunakan untuk memperbarui informasi tiket, seperti deskripsi, prioritas, atau kategori. Khusus untuk merubah status ke 'On Progress', Actor yang melakukan perubahan harus merupakan salah satu kolaborator. Endpoint ini menggunakan method PATCH dan memerlukan ID tiket sebagai path parameter. Penggunaan endpoint seperti gambar 3.11



Gambar 3.11. API untuk memperbarui data tiket

C.1.5 Delete Ticket

Digunakan untuk menghapus tiket berdasarkan ID. User dapat menulis ID tiket dalam endpoint sebagai path variable, yang akan menghapus tiket dengan ID tersebut. Penggunaan endpoint seperti gambar 3.12



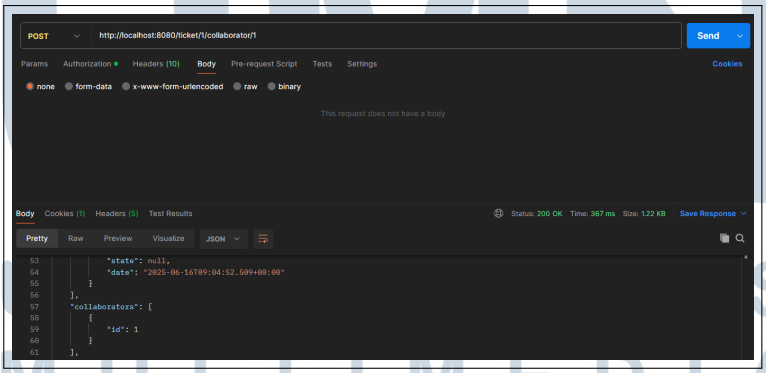
Gambar 3.12. API untuk menghapus tiket

C.2 API Collaborator

API Collaborator digunakan untuk mengelola daftar personel tambahan yang terlibat dalam proses penyelesaian tiket. *Endpoint* ini mendukung operasi penambahan dan penghapusan kolaborator dalam suatu tiket tertentu.

C.2.1 Add Collaborator

Endpoint ini digunakan untuk menambahkan satu atau beberapa kolaborator ke tiket tertentu. Data yang dikirim meliputi ID tiket dan daftar ID personel yang akan ditambahkan sebagai kolaborator. Jika data berhasil disimpan, sistem akan mengembalikan data tiket dengan daftar kolaborator terbaru yang terhubung dengan tiket tersebut. Penggunaan endpoint seperti gambar 3.13

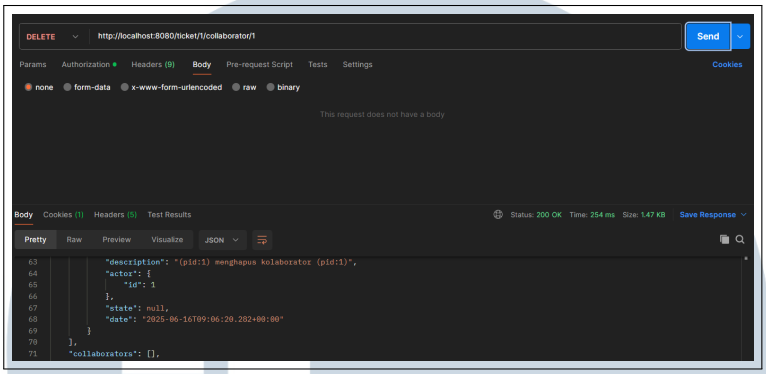


Gambar 3.13. API *Add Collaborator*

C.2.2 Remove Collaborator

Endpoint ini digunakan untuk menghapus satu kolaborator dari tiket. Permintaan dilakukan dengan menyertakan ID kolaborator sebagai path parameter.

Hanya user yang berwenang terhadap tiket yang dapat mengakses endpoint ini. Penggunaan endpoint seperti gambar 3.14



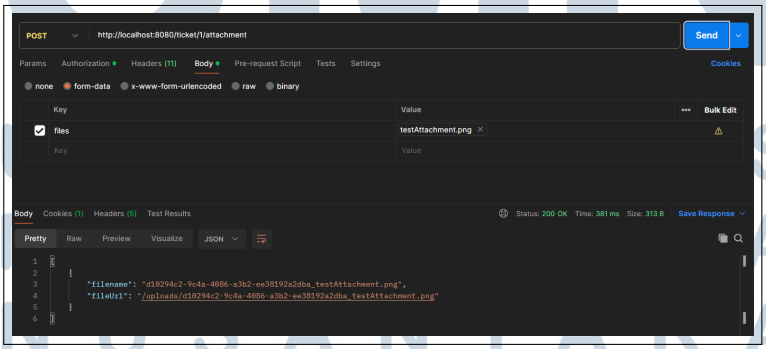
Gambar 3.14. API *Remove Collaborator*

C.3 API Attachment

API Attachment digunakan untuk menangani file yang dilampirkan dalam tiket maupun follow-up. File yang ditujukan meliputi dokumen dan gambar yang relevan dengan isi laporan tiket.

C.3.1 Upload Attachment

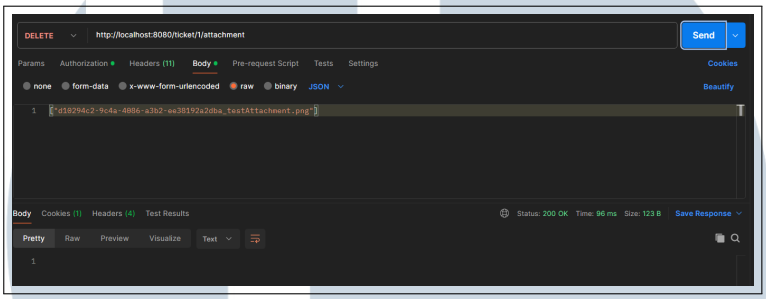
Endpoint ini menerima file melalui form-data, dengan ID tiket sebagai *path variable* dalam *endpoint*. Sistem akan menyimpan file secara fisik (lokal atau cloud) dan mencatat path serta informasi file ke dalam database. Penggunaan endpoint seperti gambar 3.15



Gambar 3.15. API *Add Attachment*

C.3.2 Delete Attachments

Endpoint ini berfungsi untuk menghapus *attachment* dari sebuat tiket. JSON *request* yang ditulis yaitu berupa *filename attachment* yang tersimpan dalam basis data. Penggunaan endpoint seperti gambar 3.16



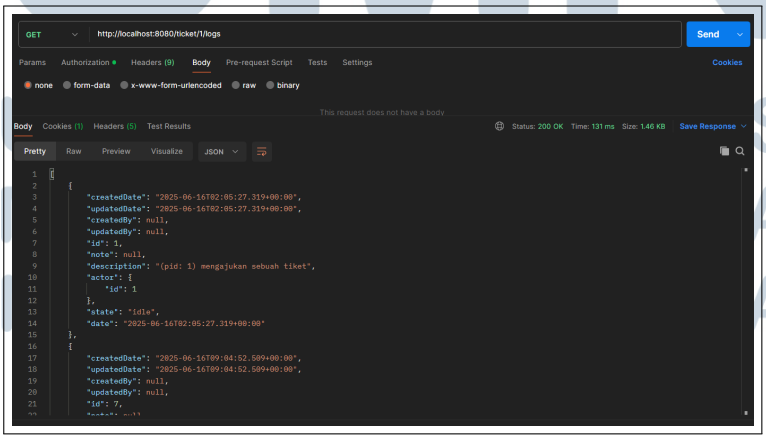
Gambar 3.16. API Delete Attachment

C.4 API Logs

API Logs memberikan akses terhadap histori aktivitas yang dilakukan pada sebuah tiket, seperti perubahan status, pengalihan divisi, atau follow-up penting. Log ini digunakan untuk keperluan monitoring dan audit sistem.

C.4.1 View Logs by Ticket

Endpoint ini digunakan untuk melihat seluruh log yang terkait dengan satu tiket tertentu. Data yang ditampilkan meliputi waktu kejadian, tipe aksi (e.g., create, update, assign), deskripsi, dan siapa yang melakukan aksi tersebut. Penggunaan endpoint seperti gambar 3.17

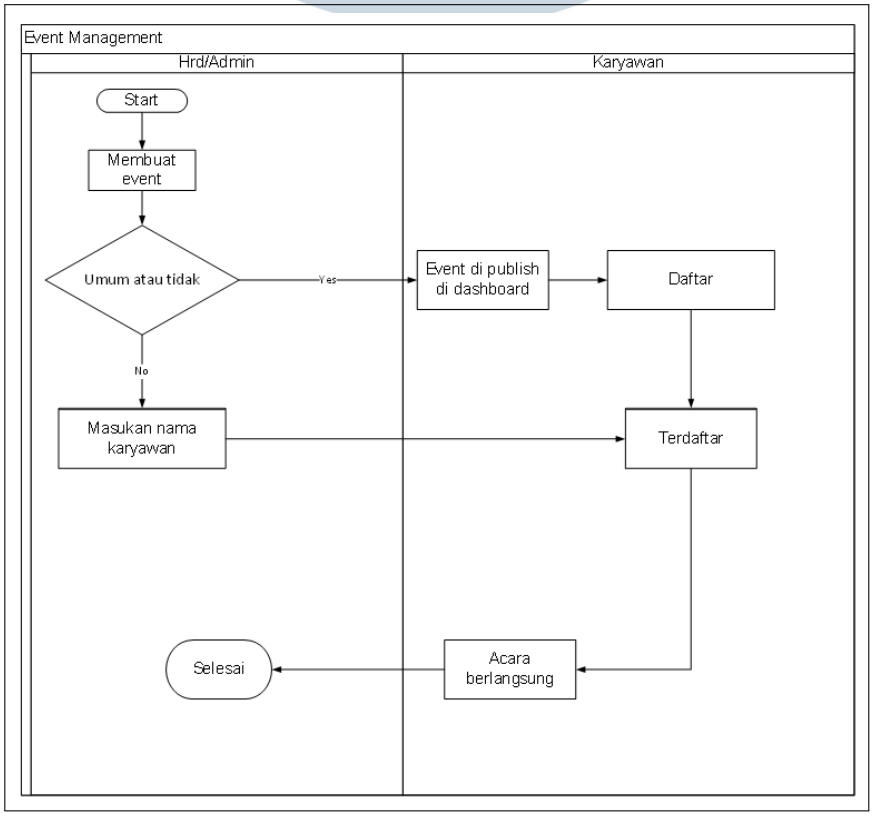


Gambar 3.17. API Ticket Logs

3.3.3 Minggu ke-4 s/d ke-6: Projek Event Management System

Pada minggu ke-4 hingga ke-6, mahasiswa berkesempatan untuk terlibat dalam pengembangan sistem *Event Management*, sebuah modul tambahan pada platform internal perusahaan yang ditujukan untuk mengelola penyelenggaraan kegiatan internal seperti seminar, workshop, pelatihan, dan acara lainnya. Infrastruktur sistem masih menggunakan stack teknologi yang sama seperti projek sebelumnya, yaitu *Java Spring Boot* sebagai *back-end framework*, *MySQL* sebagai database, serta *Postman* untuk pengujian endpoint.

Tujuan dari sistem ini adalah untuk mempermudah admin dalam membuat dan mengelola event, serta memberikan sarana bagi karyawan untuk melihat dan mendaftar pada event yang tersedia. Sistem juga mengatur jumlah peserta, lokasi acara, status pendaftaran, dan mengelola skenario penjadwalan ulang (reschedule) jika dibutuhkan. Flow utama sistem dimulai dari pembuatan event oleh admin, publikasi ke dashboard, pendaftaran oleh karyawan, dan proses reminder serta kontrol jumlah peserta, termasuk fitur waiting list jika kuota penuh. Flow dapat dilihat di gambar 3.18



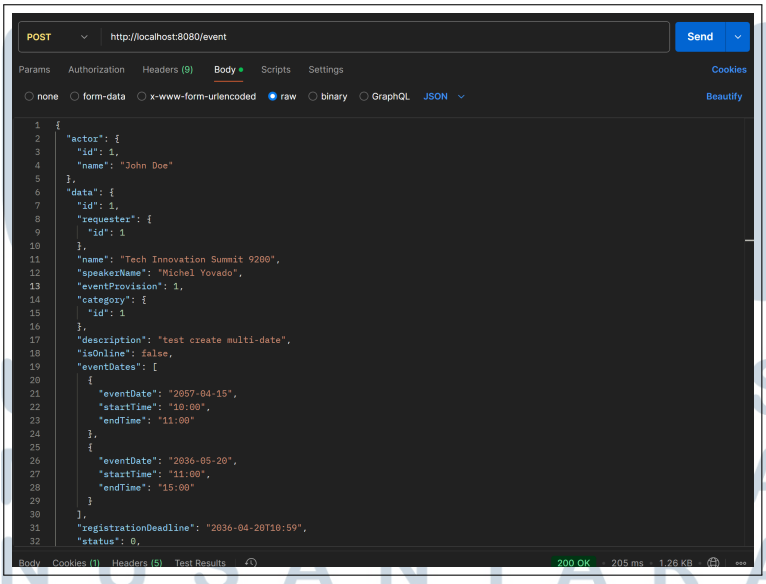
Gambar 3.18. Flow Sistem Event Management

A Pembuatan API

Sistem Event Management dibangun dengan pendekatan RESTful API. Pembuatan API mencakup endpoint untuk mengelola entitas master seperti kategori dan lokasi, yang memiliki struktur sama dengan entitas master pada projek sebelumnya. Kemudian endpoint untuk transaksi event, pendaftaran karyawan, dan penjadwalan ulang. Penjelasan mengenai API-API utama yang dikembangkan akan dijelaskan sebagai berikut.

A.1 CRUD Event

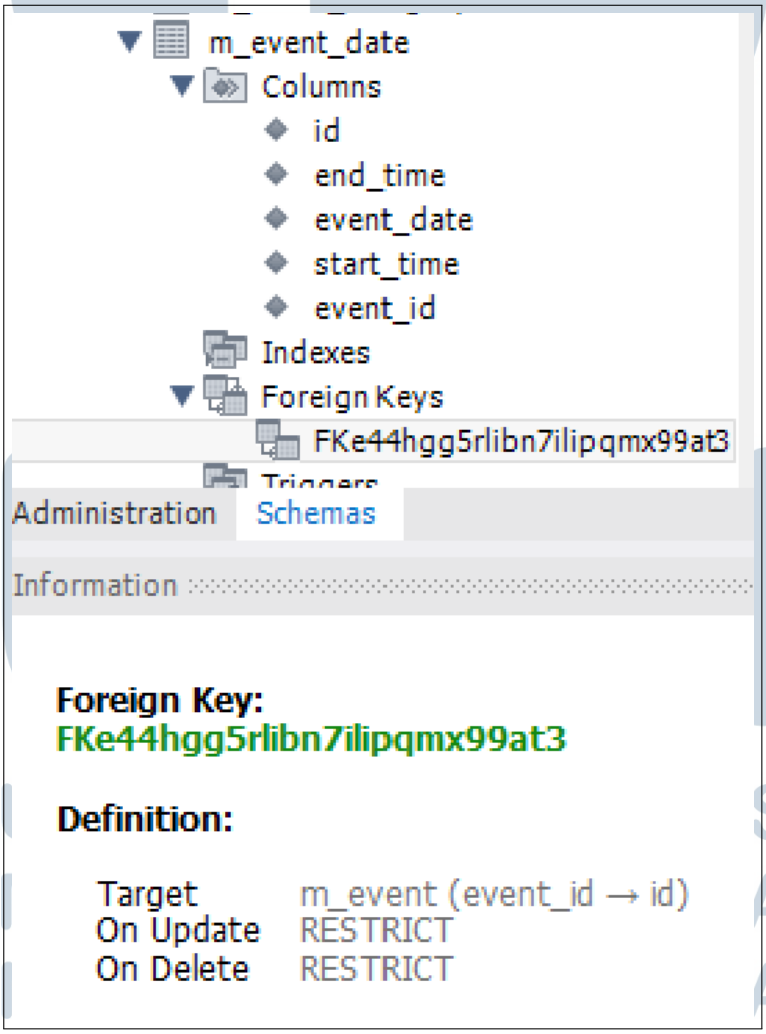
Salah satu API yang dikembangkan adalah endpoint dasar CRUD (*Create, Read, Update, Delete*) untuk mengelola data event. Admin dapat membuat event baru dengan berbagai parameter seperti nama acara, tanggal, kapasitas, pembicara, kategori, deskripsi, serta pengaturan lokasi (offline/online). Endpoint untuk `create` dan `update` dilengkapi dengan validasi jam dan tanggal, agar tidak terjadi konflik antar sesi. Kemudian terdapat juga endpoint untuk melihat seluruh event atau detail satu event berdasarkan ID. Penghapusan event juga dimungkinkan, khususnya untuk event yang statusnya masih draft. Contoh pembuatan suatu event dapat dilihat di gambar 3.19



Gambar 3.19. API Create Event

A.2 Reschedule

Dalam sistem ini, entitas *event* menggunakan struktur *multi-date*, di mana satu *event* bisa memiliki beberapa tanggal pelaksanaan. Untuk mengatur ini, dibangun tabel relasi `m_event_date` yang dapat menyimpan lebih dari satu tanggal, khususnya jam mulai dan selesai. *Endpoint reschedule* memungkinkan admin untuk memperbarui tanggal tertentu yang sudah terdaftar, atau menambahkan tanggal baru jika status event masih draft. Validasi khusus ditambahkan agar tanggal baru tidak beririsan dan tetap mengikuti batas maksimal pendaftaran. Adapun struktur tabel `m_event_date` yang memiliki relasi dengan entitas *event* dapat dilihat di gambar 3.20



Gambar 3.20. Struktur Tabel `m_event_date`

A.3 Pendaftaran Karyawan

Untuk pendaftaran karyawan ke dalam event, dikembangkan beberapa endpoint yang menggunakan `path variable` berupa ID event dan ID karyawan. Endpoint utama yang digunakan adalah:

- `POST /event/{eventID}/register/{personID}`
  Endpoint ini digunakan untuk mendaftarkan karyawan ke dalam event tertentu. Jika jumlah peserta belum mencapai kuota, maka karyawan langsung dimasukkan sebagai peserta. Namun jika kuota penuh, maka karyawan akan otomatis dimasukkan ke dalam daftar waiting list.

- `GET /event/{eventID}/registered`
  Endpoint ini digunakan untuk melihat seluruh karyawan yang telah berhasil terdaftar dalam suatu event. Data peserta akan ditampilkan dalam bentuk list JSON.

- `DELETE /event/{eventID}/cancel/{personID}`
  Endpoint ini digunakan untuk membatalkan keikutsertaan seorang karyawan dari event tertentu. Setelah melakukan pembatalan, sistem akan menandai bahwa karyawan tersebut tidak bisa mendaftar kembali ke event yang sama, sehingga menghindari pendaftaran ulang yang tidak diinginkan.

Selain pendaftaran reguler, ditambahkan fitur **waiting list** untuk mengakomodasi peserta yang tidak dapat langsung masuk karena kuota penuh. Waiting list ini dapat diakses dan dikelola melalui beberapa endpoint:

- `POST /event/waiting-list/{eventID}/add/{employeeID}`
  Endpoint ini digunakan untuk secara manual menambahkan karyawan ke daftar waiting list suatu event, misalnya jika pendaftaran dilakukan oleh admin secara langsung.

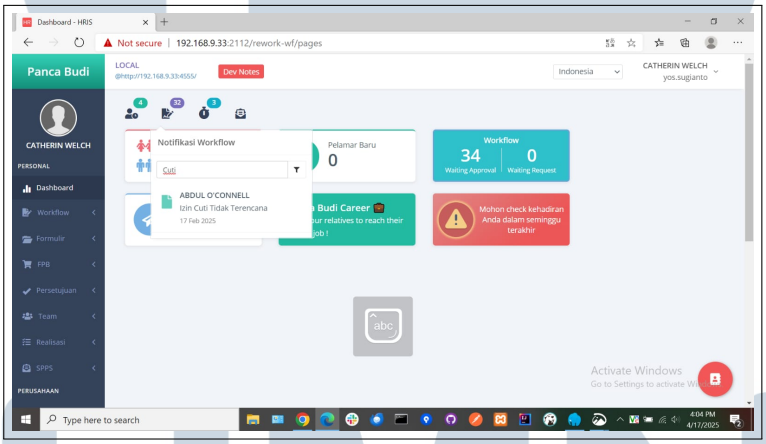
- `POST /event/waiting-list/{eventID}/approve/{employeeID}`
  Melalui endpoint ini, admin dapat menyetujui karyawan dalam waiting list untuk masuk ke daftar peserta resmi event, jika ada peserta lain yang membatalkan atau jika kuota ditambah.

- `GET /event/waiting-list/{eventID}`
  Digunakan untuk melihat daftar seluruh karyawan yang berada dalam waiting

list suatu event. Ini mempermudah admin untuk mengelola antrian dan menentukan siapa yang bisa diprioritaskan.

3.3.4 Minggu ke-7 s/d ke-9: End-user Testing untuk Aplikasi HRIS

Pada minggu ke-7 hingga ke-9, kegiatan difokuskan pada aktivitas *end-user testing* untuk aplikasi HRIS (*Human Resource Information System*), khususnya pada fitur-fitur perizinan. Tujuan dari tahap ini adalah untuk memastikan bahwa seluruh *workflow* perizinan berjalan sesuai dengan logika bisnis yang telah dirancang, serta untuk menemukan dan mendokumentasikan *bug* atau ketidaksesuaian sistem yang muncul selama pengujian. Laman aplikasi dapat dilihat di gambar 3.21



Gambar 3.21. Halaman Utama HRIS

Pengujian dilakukan terhadap berbagai jenis skenario perizinan, seperti izin sakit, izin terlambat, izin pulang cepat, cuti karena istri melahirkan, dan beberapa jenis izin lainnya yang umum diajukan oleh karyawan. Fokus utama testing adalah memvalidasi logika tanggal (termasuk pengecekan terhadap *backdate* atau tanggal lampau), ketepatan validasi antar jenis izin yang dapat bertabrakan, validitas alur pembatalan izin, serta keterhubungan pengajuan *workflow* ke halaman *approver* yang bertugas memverifikasi *workflow*.

Selain itu, dilakukan juga pengujian apakah workflow yang telah diajukan muncul pada laman *approver* yang sesuai, serta apakah notifikasi dan perubahan status izin berjalan dengan benar dari sisi pengguna maupun pihak *approver*. Kemudian juga dicek dari sisi penanggalan, misalnya ketika izin sakit diajukan bersamaan dengan izin pulang cepat dalam rentang waktu yang tumpang tindih.

Seluruh hasil pengujian dicatat dan didokumentasikan dalam platform Notion dalam bentuk tabel. Setiap entri tabel mencakup kolom langkah pengujian, ekspektasi hasil, dan hasil aktual dari sistem. Dengan pendekatan ini, proses identifikasi bug menjadi lebih sistematis dan mudah ditelusuri kembali oleh tim pengembang untuk proses perbaikan. Contoh salah satu dokumentasi testing seperti pada gambar 3.22

Gambar 3.22. Dokumentasi Testing via Notion

3.3.5 Minggu ke-10 dan ke-11: Pengenalan dan Setup Odoo

Pada minggu ke-10 dan ke-11, kegiatan difokuskan pada proses pengenalan dan setup awal pengembangan sistem menggunakan Odoo. Odoo adalah platform ERP open-source berbasis Python yang menyediakan berbagai macam modul bisnis yang saling terintegrasi, seperti modul penjualan, pengadaan, akuntansi, SDM, hingga manufaktur. Tidak seperti framework konvensional seperti Spring Boot, Odoo menggunakan arsitektur modular penuh dan menggunakan pendekatan berbasis objek, yang memungkinkan pengembangan dilakukan dengan struktur yang lebih terstandarisasi. Logo Odoo seperti gambar 3.23

Gambar 3.23. Logo Odoo

1. Tools yang Digunakan

Selama proses setup dan pengembangan awal Odoo, digunakan beberapa tools utama sebagai berikut:

- **Python 3.8**
  Sebagai bahasa dasar yang digunakan oleh Odoo, Python 3.8 dipilih karena kompatibilitasnya dengan versi Odoo yang digunakan (Odoo 14). Versi ini cukup stabil dan mendukung seluruh dependensi yang dibutuhkan oleh Odoo.
- **PostgreSQL**
  Odoo secara default menggunakan PostgreSQL sebagai sistem manajemen basis data (DBMS). Oleh karena itu, proses instalasi PostgreSQL dilakukan terlebih dahulu, kemudian dibuat user dan role database khusus untuk Odoo agar dapat menjalankan dan menyimpan data modul yang dikembangkan.
- **PyCharm Community Edition**
  PyCharm digunakan sebagai *Integrated Development Environment (IDE)* untuk menulis dan mengelola kode Python dalam modul Odoo. IDE ini digunakan karena sudah dirancang khusus untuk penulisan Python, serta dilengkapi fitur *syntax highlighting*, *auto-completion*, dan integrasi terminal yang sangat membantu dalam proses *development*.

- **Odoo Community Edition**
  Digunakan sebagai platform ERP utama, Odoo 14 diunduh dari repositori GitHub resmi Odoo dan dijalankan secara lokal. Versi ini mendukung berbagai fitur inti, serta kompatibel dengan kebutuhan sistem yang akan dikembangkan selama program magang.

2. Instalasi dan Setup Lingkungan Pengembangan

Langkah awal yang dilakukan adalah meng-clone repositori Odoo Community Edition dari GitHub. Setelah itu, dilakukan instalasi semua *dependency* menggunakan `pip` dan pengaturan konfigurasi environment. Dependensi umum seperti `psycopg2`, `babel`, `Werkzeug`, `lxml`, `Pillow`, dan `reportlab` dipastikan terpasang agar sistem dapat berjalan tanpa error.

Setelah semua paket terpasang, dibuat file konfigurasi bernama `odoo.conf` yang memuat pengaturan utama seperti nama basis data, path direktori add-ons, port yang digunakan, serta pengaturan log. Setelah selesai mengkonfigurasi pengaturan yang diperlukan, maka dapat menjalankan server yang dapat diakses melalui browser dengan alamat `localhost`.

3. Struktur Modular Odoo

Odoo menggunakan pendekatan berbasis modul untuk semua fitur yang dikembangkan. Setiap fitur (seperti *sales*, *invoice*, dan lain-lain) dibuat dalam bentuk modul yang berdiri sendiri dan dapat dimodifikasi secara langsung. Struktur standar dari satu modul Odoo terdiri dari:

- **__manifest__.py**
  File ini berfungsi sebagai deskripsi metadata dari modul. Di dalamnya didefinisikan nama modul, versi, author, dependency, dan file XML yang digunakan dalam tampilan.

- **__init__.py**
  Berfungsi sebagai entry point Python agar Odoo dapat mengenali modul ini sebagai paket Python yang valid. File ini digunakan untuk mengimpor model-model yang ada dalam folder `models`.

- **models/**
  Folder ini berisi file Python yang mendefinisikan struktur tabel dan

logika bisnis dari modul. Setiap *class* biasanya merupakan turunan dari `models.Model` dan menggunakan ORM (*Object-Relational Mapping*) milik Odoo.

- **views/**
  Folder ini berisi file XML yang mendefinisikan tampilan antarmuka pengguna (form, tree, kanban, dan lain-lain). Dalam folder ini didefinisikan form view, list view, search view, serta menu navigasi.

- **security/**
  Folder ini berisi file kontrol akses (biasanya dalam bentuk `ir.model.access.csv`) yang menentukan siapa saja yang dapat membaca, menulis, membuat, atau menghapus data pada model tertentu.

4. Aktivasi Developer Mode dan Antarmuka Odoo

Setelah berhasil masuk ke dashboard Odoo, dilakukan aktivasi *Developer Mode* melalui menu pengaturan. Developer mode memberikan akses ke fitur-fitur penting untuk pengembangan, seperti melihat `external ID`, memeriksa struktur field dalam model, dan menyesuaikan tampilan form XML secara langsung dari UI.

5. Setup PostgreSQL dan Database Odoo

PostgreSQL merupakan sistem basis data default yang digunakan oleh Odoo. Sebelum menjalankan aplikasi, dibuat user PostgreSQL bernama `odoo` dan password default `123456789` pada program Odoo dan juga pada file `odoo.conf` via PyCharm. Selain itu, dilakukan pembuatan database awal menggunakan antarmuka Odoo atau langsung via command line PostgreSQL.

3.3.6 Minggu ke-12 s/d ke-14: Projek Sales Target

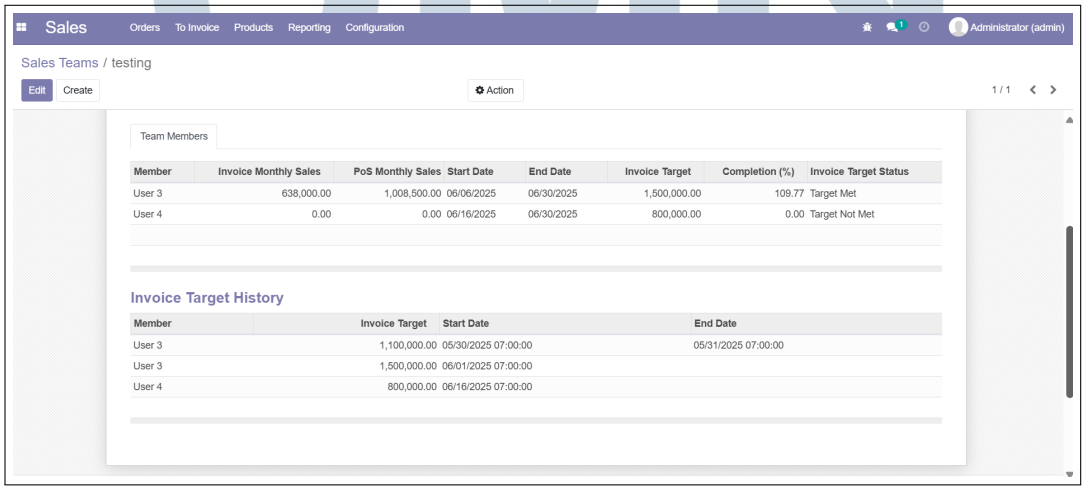
Pada minggu ke-12 hingga ke-14, fokus kegiatan diarahkan pada pengembangan modul *Sales Target* menggunakan platform Odoo. Modul ini merupakan bagian dari sistem penjualan internal yang bertujuan untuk membantu tim manajemen dalam menetapkan, memantau, dan mengevaluasi target penjualan setiap anggota tim sales, baik dalam konteks penjualan offline (*Point of Sale*) maupun online (*Quotations*).

Dalam pengembangan modul ini, digunakan infrastruktur dan tools yang sudah cukup familiar dari proyek sebelumnya. Odoo tetap menjadi platform utama yang digunakan untuk membangun dan menjalankan modul ERP, sementara proses pengkodean dilakukan menggunakan PyCharm sebagai IDE karena kemudahannya dalam menangani struktur proyek Python. Bahasa pemrograman yang digunakan adalah Python dalam model dan logika *backend* serta XLM untuk *views* dan tampilan *frontend*. Basis data dikelola menggunakan PostgreSQL, yang secara default digunakan oleh Odoo dan terintegrasi langsung dengan ORM miliknya.

Pengembangan dilakukan secara lokal, dengan server Odoo dijalankan melalui *interpreter* PyCharm menggunakan konfigurasi yang telah disesuaikan dalam file `odoo.conf`. Pengembangan fitur Sales Target dimulai dari mendefinisikan model dan field yang diperlukan, kemudian dilanjutkan dengan pembuatan tampilan (form view, list view) menggunakan XML, serta pengaturan hak akses melalui sistem security Odoo.

Tabel Hasil Sales dan *Invoice Target History*

Pada halaman tim penjualan (Sales Teams), terdapat dua bagian utama yang menjadi fitur penting dalam pengembangan modul Sales Target, yaitu tabel *Team Members* dan *Invoice Target History*. Kedua bagian ini dirancang untuk memberikan gambaran yang jelas dan langsung terhadap kinerja anggota tim dalam memenuhi target penjualan yang telah ditetapkan, baik untuk kebutuhan evaluasi jangka pendek maupun pelaporan historis. Visualisasi tabel dapat dilihat pada gambar 3.24



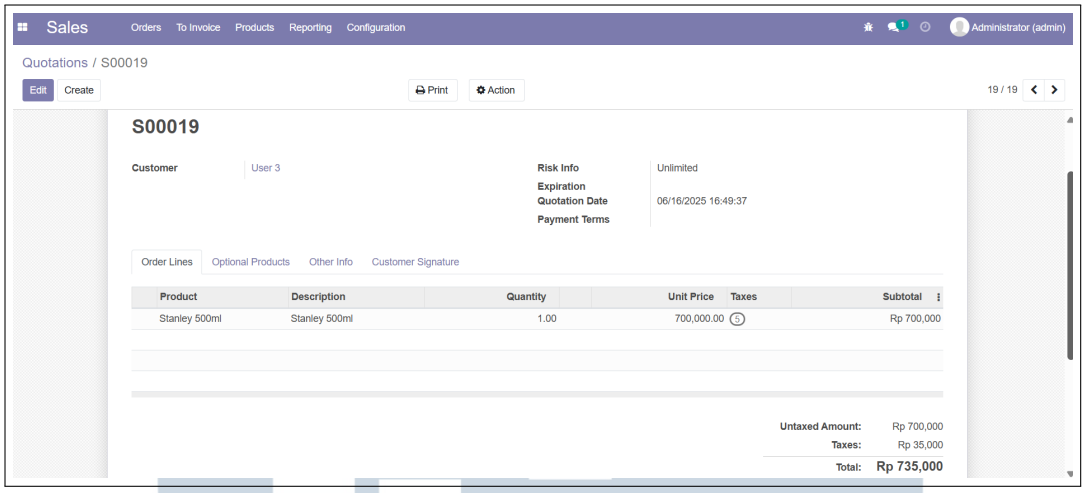
Gambar 3.24. Tampilan Halaman Sales Team

Tabel pertama, yaitu *Team Members*, menampilkan data terkini dari setiap anggota tim yang sedang berada dalam periode target aktif. Masing-masing baris mewakili seorang anggota tim, lengkap dengan informasi nama pengguna, total nilai penjualan bulanan dari *invoice*, serta total penjualan yang tercatat melalui sistem *Point of Sale* (PoS). Selain nilai penjualan, tabel ini juga menampilkan tanggal mulai dan berakhirnya periode target yang sedang berlangsung. Nilai target penjualan yang ditetapkan bagi setiap anggota ditampilkan secara berdampingan, bersama dengan persentase capaian terhadap target tersebut yang dihitung otomatis berdasarkan nilai *invoice* dan PoS. Sistem secara langsung menentukan status dari target yang dicapai, dengan memberi label "Target Met" apabila nilai penjualan telah memenuhi atau melampaui target yang ditetapkan, dan sebaliknya ditandai dengan "Target Not Met" jika belum terpenuhi. Tampilan ini tidak hanya membantu manajer penjualan untuk melakukan pemantauan performa secara real-time, tetapi juga memberikan informasi konkret yang berguna untuk mendukung pengambilan keputusan strategis, seperti evaluasi performa atau pemberian insentif.

Sementara itu, tabel kedua yang berjudul *Invoice Target History* berfungsi untuk mencatat seluruh riwayat target penjualan yang pernah dibuat untuk masing-masing anggota tim. Berbeda dengan tabel pertama yang hanya menampilkan target yang sedang aktif, tabel histori ini mencatat semua target, baik yang masih berlangsung maupun yang sudah tidak berlaku. Dalam tabel ini, setiap entri menampilkan informasi nama anggota, besaran target invoice yang ditetapkan, serta tanggal mulai dan berakhir dari masa berlaku target tersebut. Keberadaan histori ini memungkinkan tim administrasi dan manajemen untuk melakukan pelacakan dan evaluasi berkala terhadap performa individu berdasarkan periode waktu yang berbeda, serta memastikan bahwa tidak ada tumpang tindih dalam penetapan target.

Proses Pembuatan Invoice dan Integrasi ke Perhitungan Target Penjualan

Dalam proses pengembangan dan pengujian modul Sales Target, salah satu alur penting yang dilakukan mahasiswa adalah melakukan simulasi pembuatan *invoice* untuk memastikan bahwa nilai penjualan yang tercatat melalui sistem benar-benar terintegrasi dengan baik ke dalam perhitungan target bulanan anggota tim. Pembuatan *invoice* dilakukan melalui menu *Sales* di Odoo, dengan membuat *quotation* baru, mengisi data pelanggan, memilih produk, menentukan jumlah, serta harga jual, yang kemudian menjadi *invoice* yang valid dan terposting. Gambaran invoice dapat dilihat pada gambar 3.25



Gambar 3.25. Tampilan Halaman Create Invoice

Setelah *invoice* berhasil dibuat dan memiliki status yang sah (*paid*), sistem akan secara otomatis menarik nilai total dari *invoice* tersebut dan mengkalkulasikannya ke dalam field "*Invoice Monthly Sales*" pada tabel target di halaman tim penjualan. Mekanisme ini berjalan berdasarkan pencocokan data antara *user* yang melakukan penjualan dan tanggal *invoice* yang berada dalam rentang target. Nilai penjualan yang masuk kemudian dijumlahkan dan ditampilkan sebagai bagian dari progres pencapaian target masing-masing anggota tim.

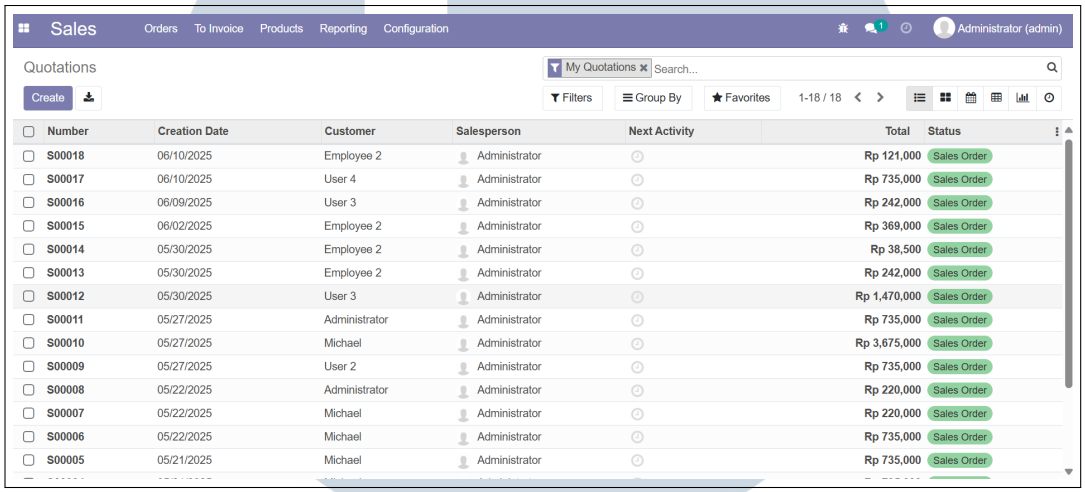
Alur ini diuji untuk memastikan bahwa setiap transaksi yang valid langsung berdampak pada laporan performa *sales* tanpa proses manual. Dengan demikian, proses pemantauan dan evaluasi target dapat berjalan secara otomatis dan akurat, yang menjadi salah satu tujuan utama pengembangan sistem ini.

Logika Tanggal dan Validasi Target Aktif

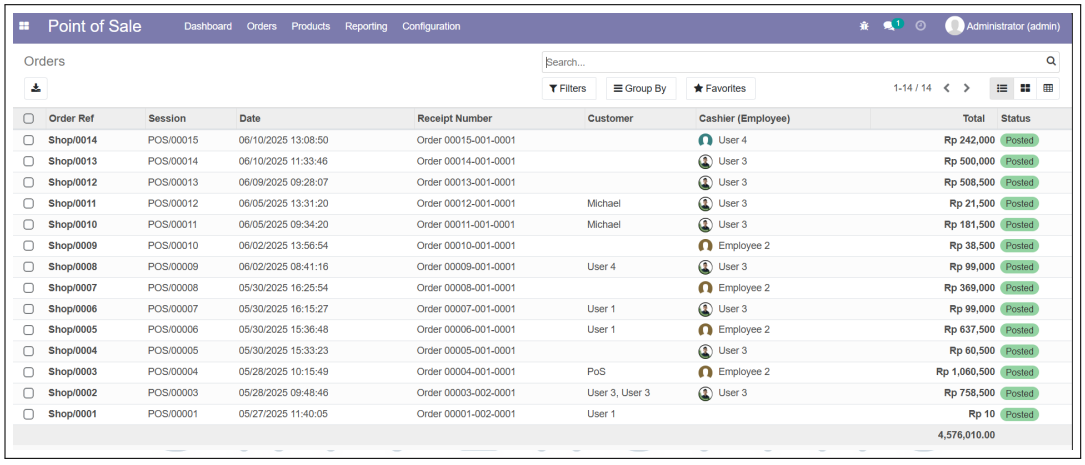
Salah satu komponen logika inti dalam sistem Sales Target adalah validasi periode waktu untuk memastikan bahwa hanya transaksi yang berada dalam rentang tanggal target yang akan dihitung dalam pencapaian penjualan. Untuk mencapai hal tersebut, setiap target yang ditetapkan untuk anggota tim penjualan memiliki informasi tanggal mulai (*start date*) dan tanggal berakhir (*end date*) yang disimpan secara eksplisit di dalam sistem. Nilai-nilai ini menjadi acuan utama dalam menentukan apakah sebuah invoice relevan terhadap target yang sedang berjalan.

Ketika sebuah invoice dibuat dan disahkan, aplikasi akan memeriksa tanggal dokumen tersebut dan mencocokkannya dengan periode aktif dari target yang dimiliki oleh user yang bersangkutan. Jika tanggal invoice berada di dalam rentang

waktu yang sesuai, maka nilainya akan diikutsertakan dalam kalkulasi pencapaian. Sebaliknya, apabila tanggal invoice berada di luar periode target—baik sebelum tanggal mulai atau setelah tanggal berakhir—maka invoice tersebut akan diabaikan dan tidak memengaruhi nilai "*Invoice Monthly Sales*" yang sedang berjalan. Hal ini dapat diverifikasi dengan melihat list penjualan anggota, seperti pada gambar 3.26 dan 3.27



Gambar 3.26. Tampilan Tabel Riwayat Quotations



Gambar 3.27. Tampilan Tabel Riwayat Point of Sale

Dengan adanya validasi ini, sistem dapat menjaga integritas data dan memberikan jaminan bahwa setiap nilai yang tercatat sebagai pencapaian memang berasal dari transaksi yang sah dan dilakukan dalam periode yang telah ditentukan. Proses ini berjalan secara otomatis di latar belakang setiap kali terjadi perubahan data pada *invoice*, baik saat pembuatan baru maupun saat dilakukan pembaruan.

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala yang Ditemukan

Selama praktik kerja magang di Panca Budi Pratama, kendala yang ditemukan mahasiswa adalah sebagai berikut:

1. **Adaptasi terhadap suasana dan lingkungan belajar baru**
   Pada awal masa magang, mahasiswa menghadapi tantangan dalam beradaptasi dengan lingkungan kerja yang baru. Berbeda dengan suasana perkuliahan yang lebih terstruktur dan teoritis, dunia kerja memiliki ritme yang lebih cepat serta menuntut inisiatif dan komunikasi yang efektif dalam tim. Mahasiswa perlu membiasakan diri untuk memahami alur kerja internal, cara berkoordinasi dengan divisi lain, dan ekspektasi kinerja yang bersifat profesional.

2. **Penggunaan infrastruktur dan platform yang belum familiar**
   Salah satu kendala teknis yang cukup signifikan adalah belum familiarnya mahasiswa dengan infrastruktur teknologi yang digunakan, seperti framework Odoo, basis data PostgreSQL, serta alur pengembangan sistem berbasis modular. Sebelumnya, mahasiswa lebih terbiasa menggunakan platform berbasis Java dan MySQL. Perbedaan struktur file, cara kerja ORM, serta pendekatan Odoo yang unik membutuhkan waktu tambahan untuk dipelajari dan dikuasai.

3. **Keterbatasan waktu dalam mengerjakan proyek dan tugas**
   Karena banyaknya modul dan fitur yang dikerjakan dalam waktu yang cukup terbatas, mahasiswa perlu membagi waktu secara efektif untuk menyelesaikan tugas utama, melakukan debugging, dan menjalankan pengujian sistem. Di beberapa kesempatan, mahasiswa harus menyelesaikan pengembangan modul sambil tetap mendokumentasikan progres serta menjalani kegiatan tambahan seperti meeting, revisi fitur, dan testing.

3.4.2 Solusi yang Digunakan

Dalam rangka mengatasi kendala-kendala yang ditulis di atas, ada beberapa pendekatan yang dilakukan oleh mahasiswa:

1. **Meningkatkan komunikasi dan keterbukaan terhadap tim**

   Untuk mengatasi kesulitan adaptasi lingkungan kerja, mahasiswa berupaya aktif berdiskusi dengan rekan satu divisi dan pembimbing lapangan. Dengan bertanya langsung saat mengalami kebingungan serta mengikuti ritme kerja tim secara bertahap, mahasiswa menjadi lebih terbiasa dan percaya diri dalam menyelesaikan tanggung jawab yang diberikan.

2. **Melakukan eksplorasi mandiri dan praktik langsung terhadap teknologi baru**

   Untuk memahami Odoo dan teknologi lain yang belum familiar, mahasiswa meluangkan waktu di luar jam kerja untuk membaca dokumentasi resmi, mengikuti tutorial, serta mencoba membuat modul sederhana secara mandiri. Pendekatan ini membantu mahasiswa mempercepat proses belajar dan lebih memahami bagaimana alur kerja Odoo dari sisi back-end dan UI.

3. **Membuat perencanaan mingguan dan memprioritaskan pekerjaan utama**

   Dalam menghadapi tekanan waktu, mahasiswa menyusun daftar pekerjaan berdasarkan prioritas dan mendiskusikannya bersama pembimbing atau user. Dengan fokus pada pekerjaan yang paling krusial terlebih dahulu, proses pengerjaan proyek dapat lebih terarah dan hasilnya pun dapat disesuaikan dengan ekspektasi perusahaan tanpa mengorbankan kualitas.