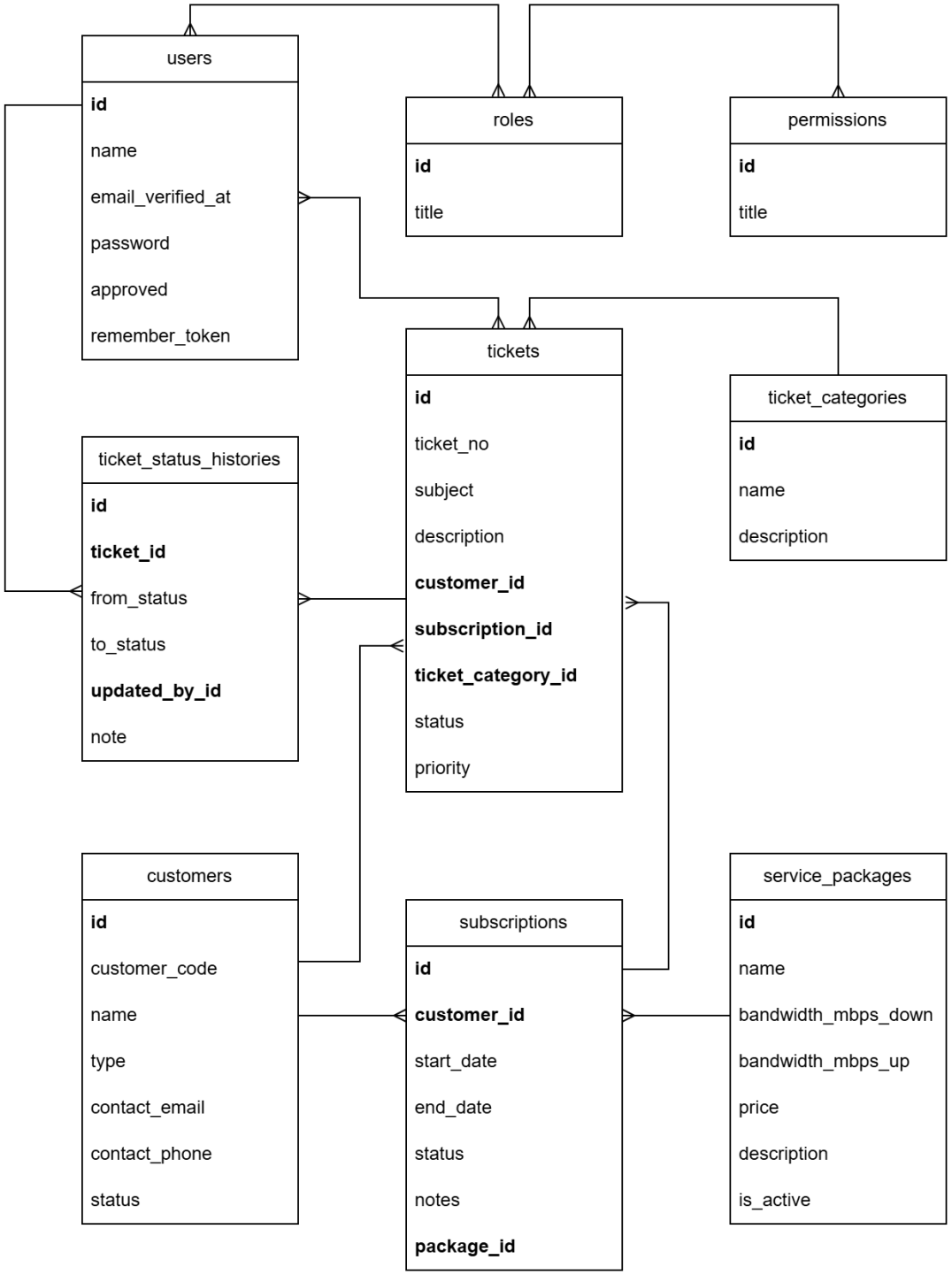
1 Perancangan Sistem dan Basis Data

1.1 Entity-Relationship Diagram (ERD)



1. Manajemen Pengguna dan Hak Akses:

· Users: Tabel ini menyimpan data pengguna yang dapat mengakses sistem, seperti Admin, Agent NOC, dan Customer Service.

· Roles: Mendefinisikan peran-peran yang ada di dalam sistem (misalnya, 'Admin', 'Agent NOC').

· Permissions: Berisi daftar semua hak akses yang tersedia.

· Relasi: Hubungan antara users, roles, dan permissions adalah Many-to-Many.

    · Seorang User bisa memiliki banyak Role, dihubungkan oleh tabel pivot role_user.

    · Satu Role bisa memiliki banyak Permission, dihubungkan oleh tabel pivot permission_role.

2. Manajemen Pelanggan dan Layanan:

· Customers: Tabel inti yang menyimpan semua data pelanggan.

· Service Packages: Berisi daftar paket layanan internet yang ditawarkan.

· Subscriptions: Tabel ini menghubungkan pelanggan dengan paket layanan yang mereka pilih.

· Relasi:

    · Seorang Customer dapat memiliki banyak Subscriptions (One-to-Many). Ini memungkinkan satu pelanggan mendaftarkan beberapa layanan di lokasi berbeda.

    · Setiap Subscription terhubung ke satu ServicePackage (Many-to-One).

3. Manajemen Tiket Gangguan:

· Tickets: Tabel utama untuk mencatat semua laporan gangguan dari pelanggan.

· Ticket Categories: Mengelompokkan jenis gangguan.

· Ticket Status Histories: Mencatat setiap perubahan status pada sebuah tiket untuk keperluan audit dan pelacakan.

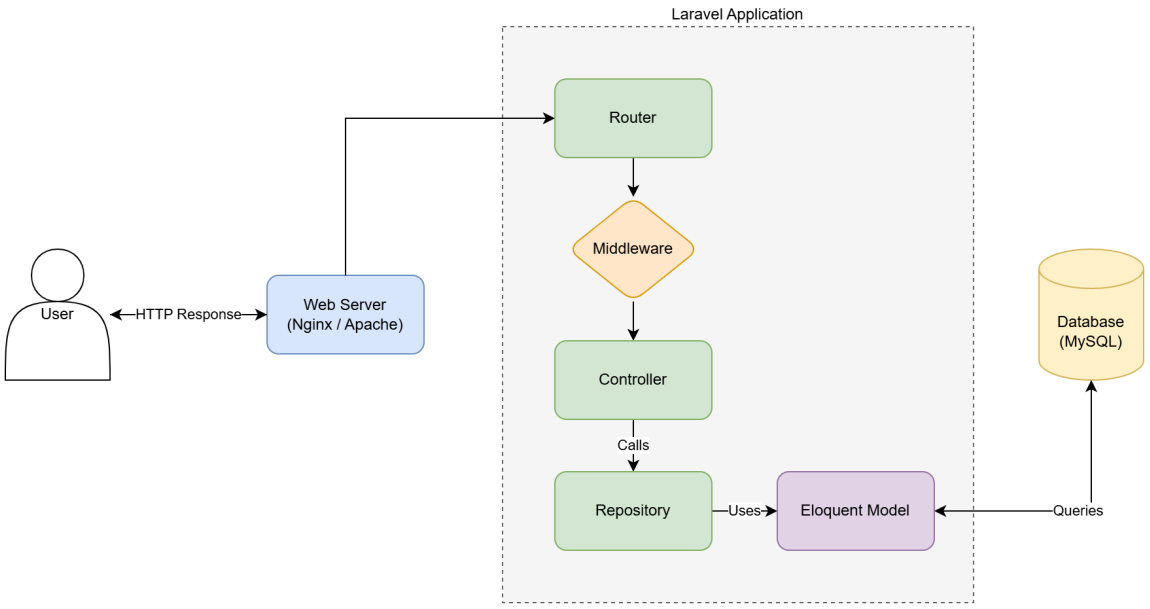
· Relasi:

  · Sebuah Ticket harus terhubung ke satu Customer dan satu Subscription (Many-to-One).

  · Sebuah Ticket juga ditugaskan ke banyak Users (agent), membentuk hubungan Many-to-Many melalui tabel pivot ticket_user.

  · Satu Ticket memiliki banyak TicketStatusHistories (One-to-Many), yang menciptakan rekam jejak (timeline) perubahan status.

1.2 Arsitektur Sistem



Penjelasan Alur Kerja dan Komponen:

Alur kerja sebuah permintaan (request) dari pengguna hingga mendapatkan respons (response) melewati beberapa komponen utama:

1. User Browser: Pengguna melakukan interaksi melalui antarmuka web atau api.

2. Web Server (Nginx/Apache): Bertindak sebagai gerbang utama yang menerima semua permintaan HTTP dari pengguna dan meneruskannya ke aplikasi ExpressJs.

3. Aplikasi Express:

    · Route: Komponen pertama yang menerima permintaan. Router memetakan URL ke method Controller yang sesuai.

    · Middleware: Lapisan keamanan yang memfilter permintaan sebelum sampai ke Controller. Ini digunakan untuk otentikasi (memastikan pengguna sudah login) dan otorisasi (memeriksa hak akses/peran pengguna).

    · Controller: Bertindak sebagai orkestrator. Controller menerima permintaan, memvalidasi input, dan memanggil logika bisnis yang relevan yang ada di dalam Repository.

    · Repository: Lapisan abstraksi antara Controller dan Model. Semua logika yang berkaitan dengan query dan manipulasi data ditempatkan di sini. Penggunaan Repository Pattern membuat Controller lebih bersih dan tidak terikat langsung dengan implementasi database, sehingga lebih mudah untuk diuji (testing) dan dikelola.

    · Model (Eloquent): Merepresentasikan tabel di dalam database. Model bertanggung jawab untuk berinteraksi langsung dengan database untuk mengambil, menyimpan, atau memperbarui data.

4. Database (MySQL): Tempat semua data aplikasi disimpan secara persisten.

2.2 Database

MySQL dipilih karena keandalannya, performa yang baik untuk aplikasi web pada umumnya, dan kompatibilitasnya yang tinggi dengan PHP dan ExpressJs. Komunitasnya yang besar juga.

3 Implementasi Fitur Utama

2. Otentikasi dan Manajemen Peran (Roles & Permissions)

Login Menggunakan sistem otentikasi bawaan ExpressJs,

Manajement permission diinisialisasi dalam auth service provider yang sudah tersimpan didalam database.

3. Demo API Singkat (Opsional):

4 Aspek Keamanan & Validasi

4.1 Proteksi Endpoint dengan Middleware:

· Otentikasi: Setiap route di dalam panel admin dilindungi oleh middleware auth bawaan ExpressJs. Ini memastikan hanya pengguna yang telah berhasil login yang dapat mengaksesnya.

· Otorisasi: Akses ke setiap modul (CRUD) dibatasi lebih lanjut menggunakan middleware

4.2 Validasi Input

Semua data yang masuk dari formulir (HTTP Request) divalidasi secara ketat menggunakan Form Request Validation ExpressJs.

4.3 Manajemen Token/Sesi

· Untuk Aplikasi Web: Menggunakan sistem sesi berbasis cookie bawaan ExpressJs.

· Untuk API: Otentikasi untuk endpoint API dijamin oleh ExpressJs. Ketika pengguna login melalui API. Token ini harus disertakan dalam header Authorization pada setiap permintaan ke API untuk verifikasi.

5 Database dan Query

Database dan query yang sudah dibuat terdapat di
https://github.com/loscardos/Resolvy/tree/master/database

6 Dokumentasi & Testing

6.1 Struktur dan Isi README.md

https://github.com/loscardos/Resolvy/blob/master/README.md

6.2 Pengujian API (API Testing)

Menggunakan SwaggerUI, pengujian dapat diakses didalam /api/documentation

6.3 Error Handling:

Response API yang diterima menggunakan struktur yang sama karena menggunakan Trait yang sama, error yang diterima oleh user akan menampilkan status code yang sesuai dengan apa yang terjadi di sisi backend.