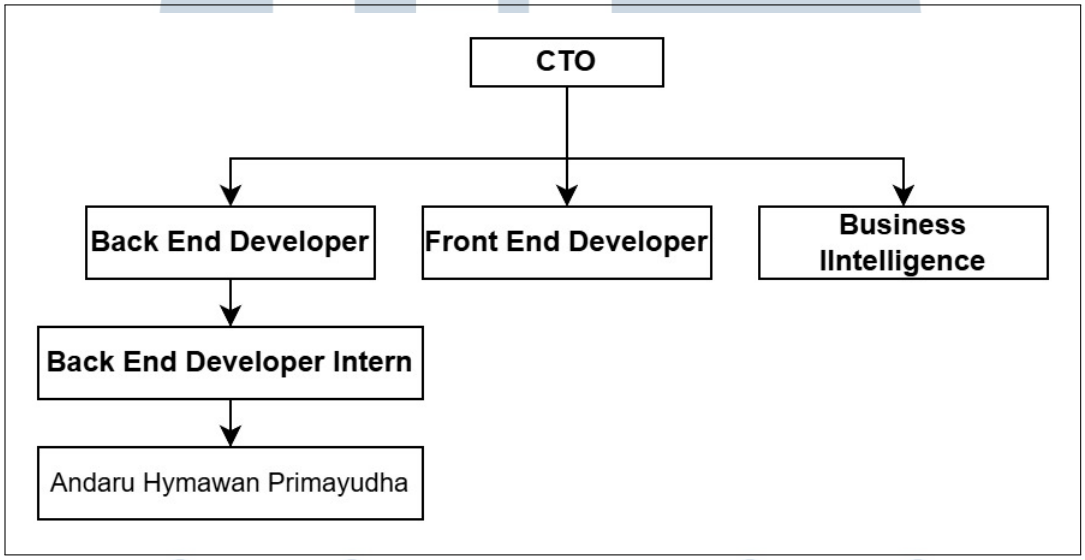
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Program kerja magang di PT Jaya Santoso Teknologi dilaksanakan dalam tim pengembang website *e-invitation* Minyma, dengan peran sebagai *back-end developer intern*.



Gambar 3.1. Struktur IT Development PT Jaya Santoso Teknologi

Sumber: [4]

Seluruh anggota divisi yang berada di bawah naungan IT Development diharapkan untuk berkoordinasi secara aktif satu sama lain guna menghindari miskomunikasi dalam pengerjaan proyek. Apabila terdapat kendala atau hal yang perlu diklarifikasi, peserta magang dapat langsung berkonsultasi dengan pembimbing yang ditugaskan.

Selama masa magang, seluruh aktivitas dan perkembangan tim *back-end* berada di bawah koordinasi langsung supervisor, Jesivinica Christy Santoso.

3.2 Tugas yang Dilakukan

Selama pelaksanaan kerja magang sebagai *back-end developer intern* di PT Jaya Santoso Teknologi, tugas utama yang dilakukan adalah perancangan dan

pembangunan sistem *backend* untuk website *e-invitation*. Pengembangan sistem ini dilakukan dari awal, mencakup pembuatan arsitektur layanan, desain basis data, implementasi logika bisnis, hingga pembuatan *Application Programming Interface* (API) yang akan digunakan oleh tim *front-end developer*.

Secara garis besar, pengembangan sistem *backend* ini terbagi menjadi beberapa modul layanan utama. Pengembangan tersebut meliputi pembuatan layanan manajemen pengguna (*user service*), yang bertanggung jawab atas pengelolaan data pengguna (CRUD), serta sistem autentikasi dan otorisasi. Selain itu, dikembangkan pula layanan manajemen acara (*event service*) dan layanan manajemen tamu (*guest service*), yang masing-masing mencakup fungsionalitas CRUD untuk entitas terkait. Beberapa fitur kunci yang dirancang dan diimplementasikan antara lain adalah sistem autentikasi pengguna yang aman serta sistem pendataan tamu (RSVP) yang komprehensif. Sistem autentikasi mencakup mekanisme login menggunakan JSON Web Token (JWT), sedangkan fitur RSVP menangani logika konfirmasi kehadiran tamu hingga pembuatan data unik berupa kode QR sebagai tiket akses masuk ke acara.

Selain pengembangan fitur utama, tugas yang dilakukan juga mencakup penerapan praktik-praktik pengembangan *backend* yang baik. Hal ini meliputi perancangan API berdasarkan prinsip RESTful, pembuatan skema basis data pada PostgreSQL, serta implementasi detail teknis seperti mekanisme *pagination*, format respon API yang standar, dan perancangan *custom exception* untuk penanganan kesalahan yang lebih spesifik.

3.3 Uraian Pelaksanaan Magang

Berikut adalah uraian pelaksanaan magang di PT Jaya Santoso Teknologisetiap minggunya yang dapat dilihat melalui 3.1

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Pemberian arahan dari pembimbing mengenai projek yang akan dikerjakan, menginstal software yang dibutuhkan, dan mempelajari tech stack yang diperlukan
2	Melanjutkan belajar mengenai Spring Boot dan mulai mengerjakan *user service*

Minggu Ke -	Pekerjaan yang dilakukan
3	Membuat dan memperbaiki *model*, *service*, dan *repository* pada *user service*, serta mempelajari Spring Security
4	Mempelajari dan mengimplementasikan Spring Security, mulai mengimplementasikan fitur login dengan SSO dan pengamanan *endpoint* menggunakan JWT
5	Mengimplementasikan manajemen role dan permission pengguna, mulai mengimplementasikan session dan cookie untuk keperluan autentikasi
6	Mengubah alur kode dan merevisi beberapa kode karena ada penyesuaian
7	Deploy kode ke Heroku dan mulai mengimplementasikan migrasi database menggunakan Flyway
8	Membuat CRUD *role* dan *permission* pada *internal dashboard* dan membuat API tambahan untuk *payment service*
9	Mulai mendokumentasikan kode dan semua *endpoint* di Postman, serta mempelajari tentang *microservices*
10	Mengimplementasikan *microservices* pada *dummy* projek
11	Merancang flow dan basis data untuk *event* dan *guest service*, serta memperbaiki kode yang telah dideploy sebelumnya
12	Melanjutkan pembuatan *controller* dan *service* pada *guest* dan *event service*, serta mulai merancang sistem RSVP
13	Integrasi API autentikasi dengan tim *front-end*
14	Melanjutkan sistem RSVP, implementasi ulang Spring Security, dan membuat *converter* file .csv/.xls ke daftar tamu
15	Memantau integrasi API oleh tim *front-end* dan merombak sistem autentikasi dari *cookie* ke *Header*
16	Melanjutkan perombakan sistem autentikasi dan membuat sistem kode QR untuk RSVP
17	Melakukan perombakan alur sistem RSVP, mendokumentasikan API menggunakan Swagger, melakukan refaktorisasiisasi kode agar lebih *reusable*, serta melakukan *reset* basis data di Heroku
18	Membuat paginasi untuk API, migrasi dari Model Mapper ke MapStruct, dan memperbarui *authentication filter*

Minggu Ke -	Pekerjaan yang dilakukan
19	Memisahkan *guest service* untuk pengguna dan admin, membuat *controller* admin untuk *internal dashboard*, dan memperbarui login menggunakan OAuth2 Google

3.3.1 Software yang Digunakan

Berikut beberapa *software* yang digunakan dalam proses perancangan *backend* pada website *e-invitation* Minyma selama pelaksanaan program magang:

1. *Framework* Java Spring Boot

   Spring Boot adalah kerangka kerja berbasis Java yang mempermudah pengembangan aplikasi Spring dengan menyediakan konfigurasi default dan fitur siap pakai[5]. Hal ini memungkinkan pembuatan aplikasi mandiri yang dapat langsung dijalankan dengan minimal konfigurasi.

2. Intellij IDEA Community

   IntelliJ IDEA merupakan *Integrated Development Environment* (IDE) yang mendukung berbagai bahasa pemrograman, terutama Java[6]. Selama magang, digunakan versi *Community Edition* yang gratis dan menyediakan fitur esensial untuk pengembangan aplikasi.

3. Postman

   Postman adalah platform yang digunakan untuk pengembangan, pengujian, dan dokumentasi API[7]. Dengan antarmuka yang intuitif, Postman memudahkan pengembang dalam membuat dan mengelola permintaan HTTP untuk menguji *endpoint* API.

4. Docker

   Docker adalah platform yang memungkinkan pembuatan, pengiriman, dan menjalankan aplikasi dalam kontainer[8]. Dengan Docker, aplikasi beserta dependensinya dapat dikemas dalam satu unit yang konsisten dan portabel, memudahkan proses deployment di berbagai lingkungan.

5. pgAdmin

pgAdmin adalah alat administrasi dan pengembangan open-source untuk PostgreSQL, yang menyediakan antarmuka grafis untuk mengelola basis data PostgreSQL[9].

6. Github

   GitHub adalah platform berbasis cloud untuk menyimpan, berbagi, dan berkolaborasi dalam penulisan kode[10]. Dengan GitHub, pengembang dapat melacak perubahan kode, mengelola versi, dan bekerja sama dalam proyek perangkat lunak.

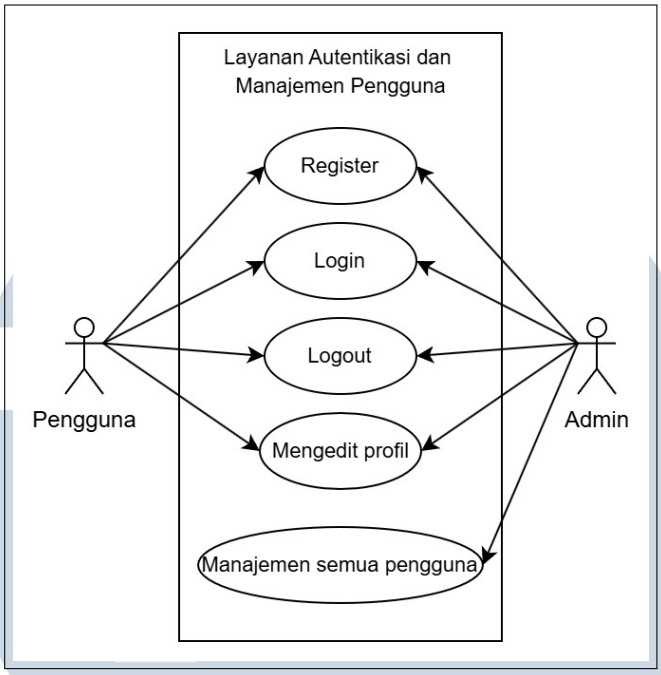
3.3.2 Perancangan Sistem dan Implementasi

A Perancangan Fungsional (Use Case Diagram)

Perancangan fungsional sistem dilakukan untuk memetakan kebutuhan serta interaksi antara aktor dengan fitur-fitur yang disediakan oleh sistem *e-invitation*. Pemodelan ini menggunakan *Use Case Diagram* guna memberikan gambaran tingkat tinggi mengenai cakupan fungsionalitas utama dalam sistem. Secara umum, fungsionalitas sistem terbagi ke dalam dua domain utama: autentikasi dan manajemen pengguna, serta manajemen acara dan tamu.

A.1 Fungsionalitas Autentikasi dan Manajemen Pengguna

Interaksi dalam domain ini melibatkan dua aktor utama, yaitu `Pengguna` dan `Admin`. Seperti yang diilustrasikan pada Gambar 3.2, aktor `Pengguna` memiliki tujuan fungsional dasar untuk mengelola sesi dan informasi pribadinya, mencakup proses `Register`, `Login`, `Logout`, dan `Edit Profil`.
Pada diagram tersebut, aktor `Admin` memiliki hubungan generalisasi terhadap aktor `Pengguna`. Artinya, `Admin` mewarisi seluruh fungsionalitas yang dimiliki `Pengguna`, dan memiliki tambahan kewenangan khusus, yaitu `Manajemen Semua Pengguna`, yang meliputi operasi *Create, Read, Update,* dan *Delete* (CRUD) terhadap data pengguna dalam sistem.



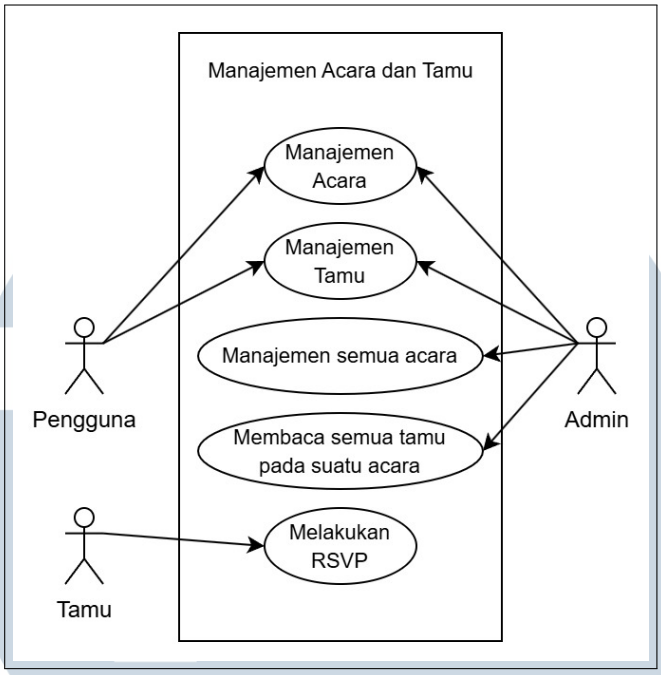
Gambar 3.2. Use Case Diagram Layanan Autentikasi dan Manajemen Pengguna

A.2 Fungsionalitas Manajemen Acara dan Tamu

Pada domain bisnis utama aplikasi, diperkenalkan aktor ketiga, yaitu `Tamu`, seperti ditunjukkan pada Gambar 3.3. Dalam konteks ini, `Pengguna` bertindak sebagai penyelenggara acara, dan memiliki fungsionalitas manajemen acara dan manajemen tamu, yang terbatas hanya pada data yang dibuat olehnya sendiri.

Sebagai turunan dari `Pengguna`, `Admin` juga memiliki hak akses terhadap fungsionalitas tersebut. Namun, `Admin` dilengkapi dengan kapabilitas tambahan berupa manajemen semua acara dan membaca semua tamu yang ada pada suatu acara, yang memungkinkan pengelolaan lintas pengguna.

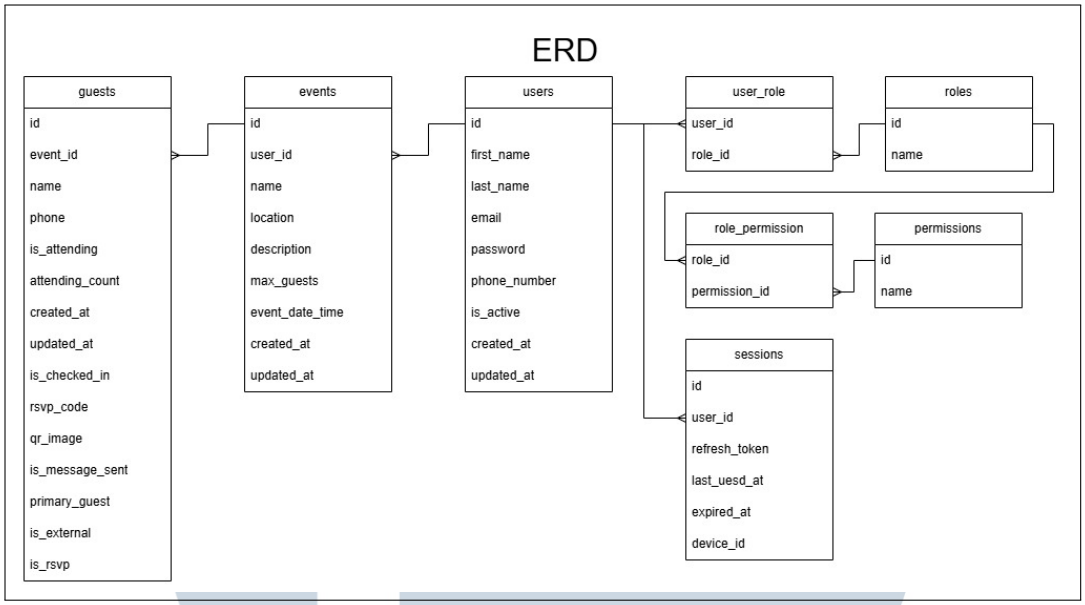
Sementara itu, aktor Tamu memiliki satu tujuan yang sangat spesifik, yaitu melakukan RSVP sebagai bentuk konfirmasi kehadiran terhadap undangan yang diterima.



Gambar 3.3. Use Case Diagram Manajemen Acara dan Tamu

B Perancangan Basis Data (ERD)

Perancangan basis data untuk sistem *e-invitation* ini menggunakan sistem manajemen basis data relasional PostgreSQL. Struktur data dirancang untuk mendukung seluruh kebutuhan fungsional aplikasi, mulai dari manajemen pengguna dan acara, hingga mekanisme autentikasi dan RSVP. Struktur dan relasi antar entitas data secara keseluruhan dapat dilihat pada Gambar 3.4.



Gambar 3.4. ERD Sistem E-invitation

Entitas inti dari sistem ini terdiri dari tabel `users`, `events`, dan `guests`. Tabel `users` berfungsi sebagai pusat data pengguna, sementara tabel `events` menyimpan informasi acara yang dibuat oleh pengguna, dan tabel `guests` mencatat data tamu untuk setiap acara. Relasi *one-to-many* antara `users` ke `events` dan `events` ke `guests` membentuk alur dasar data.

Pemilihan PostgreSQL sebagai sistem manajemen basis data didasarkan pada kemampuannya menjaga keaslian dan kekonsistenan data, dukungannya terhadap berbagai jenis data, serta kemudahannya untuk dikembangkan. Desain skema database secara keseluruhan dibuat dengan memecah informasi ke dalam beberapa tabel yang terpisah, seperti `users`, `events`, dan `guests`. Pendekatan ini bertujuan untuk mengurangi pengulangan data dan menjaga agar data tetap akurat saat ada perubahan. Penggunaan kunci utama (*Primary Key*) dan kunci asing (*Foreign Key*) pada setiap tabel memastikan bahwa hubungan antar data selalu valid, yang penting untuk ketepatan informasi acara dan tamu.

Selain itu, sistem ini mengimplementasikan model *Role-Based Access Control* (RBAC) melalui tabel `roles` dan `permissions`, yang dihubungkan oleh tabel penghubung `user_role` dan `role_permission` untuk menangani relasi *many-to-many*. Untuk manajemen sesi dan autentikasi berbasis token, tabel `sessions` digunakan, yang berelasi dengan tabel `users`. Desain basis data ini menjadi fondasi utama bagi seluruh layanan *backend* yang dibangun dalam sistem.

Terdapat alternatif desain skema database yang lain, seperti penggabungan

semua data dalam satu tabel (monolitik/denormalisasi), yang mungkin tampak menyederhanakan struktur. Namun, pendekatan ini cenderung menyebabkan pengulangan data yang tinggi, sehingga dapat mengganggu keaslian data dan menyulitkan pemeliharaan sistem. Ada pula pendekatan database non-relasional (NoSQL) yang menawarkan kelenturan lebih tinggi dalam menyimpan data, tetapi kurang direkomendasikan untuk proyek ini karena kebutuhan yang kuat akan hubungan dan konsistensi data yang ketat antar entitas `users`, `events`, dan `guests`. Oleh karena itu, skema relasional dengan pemecahan data ke dalam tabel-tabel yang logis menyeimbangkan kebutuhan akan keaslian data yang tinggi, konsistensi, kemudahan pemeliharaan, dan kelenturan untuk penambahan fitur di masa depan, menjadikannya pilihan optimal untuk platform Minyma.

Untuk memberikan pemahaman yang lebih mendalam mengenai struktur basis data yang dirancang, bagian ini akan menguraikan setiap entitas (tabel) secara spesifik. Setiap penjelasan tabel akan mencakup tujuan, daftar atribut (kolom) beserta tipe data dan perannya (seperti *Primary Key*, *Foreign Key*, dan *Unique Constraint*).

B.1 Tabel `users`

Tabel users berfungsi sebagai pusat penyimpanan informasi dasar mengenai setiap pengguna terdaftar dalam sistem. Tabel ini merepresentasikan entitas pengguna utama yang datanya direferensikan oleh tabel `events`, `sessions`, dan `user_role` untuk mengaitkan acara, sesi, dan peran dengan pengguna tertentu. Rincian struktur kolom tabel ini dapat dilihat pada Tabel 3.2.

Tabel 3.2. Struktur Tabel `users`

Nama Kolom	Tipe Data	Constraint
id	INTEGER	PRIMARY KEY, NOT NULL
first_name	VARCHAR	NOT NULL
last_name	VARCHAR	-
email	VARCHAR	UNIQUE, NOT NULL
password	VARCHAR	-
phone_number	VARCHAR	DEFAULT '-'
address	VARCHAR	DEFAULT '-'
is_active	BOOLEAN	DEFAULT TRUE
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Berikut merupakan penjelasan setiap atribut pada tabel `users`:

1. `id`: Pengidentifikasi unik untuk setiap pengguna. Kolom ini berfungsi sebagai kunci utama tabel dan tidak boleh kosong.
2. `first_name`: Nama depan pengguna. Kolom ini tidak boleh kosong.
3. `last_name`: Nama belakang pengguna.
4. `email`: Alamat email pengguna, yang juga berfungsi sebagai username untuk login. Nilai pada kolom ini harus unik dan tidak boleh kosong.
5. `password`: Password pengguna yang telah di-hash untuk tujuan keamanan.
6. `phone_number`: Nomor telepon kontak pengguna. Jika tidak diisi, nilai defaultnya adalah '-'.
7. `address`: Alamat tempat tinggal pengguna. Jika tidak diisi, nilai defaultnya adalah '-'.
8. `is_active`: Status boolean yang menunjukkan apakah akun pengguna aktif atau tidak. Jika tidak diisi, nilai defaultnya adalah TRUE.
9. `created_at`: Tanggal dan waktu akun pengguna dibuat. Jika tidak diisi, nilai defaultnya adalah waktu saat ini.
10. `updated_at`: Tanggal dan waktu terakhir data pengguna diperbarui. Jika tidak diisi, nilai defaultnya adalah waktu saat ini.

B.2 Tabel `roles`

Tabel roles menyimpan definisi peran-peran yang ada dalam sistem, seperti "Admin" atau "User" biasa, sebagai bagian dari implementasi *Role-Based Access Control* (RBAC). Peran-peran ini kemudian direferensikan oleh tabel `user_role` dan `role_permission`. Rincian struktur kolom tabel ini dapat dilihat pada Tabel 3.3.

Tabel 3.3. Struktur Tabel `roles`

Nama Kolom	Tipe Data	Constraint
id	INTEGER	PRIMARY KEY, NOT NULL
name	VARCHAR	UNIQUE, NOT NULL

Berikut merupakan penjelasan setiap atribut pada tabel `roles`:

1. `id`: Pengidentifikasi unik untuk setiap peran. Kolom ini berfungsi sebagai kunci utama tabel dan tidak boleh kosong.
2. `name`: Nama unik dari peran (misalnya ”Admin”, ”User”). Nilai pada kolom ini harus unik dan tidak boleh kosong.

B.3 Tabel `user_role`

Tabel `user_role` adalah tabel penghubung yang mereferensikan `id` dari tabel `users` dan `id` dari tabel `roles`, berfungsi untuk mendefinisikan relasi antara pengguna dan peran. Relasi ini memungkinkan satu pengguna untuk memiliki lebih dari satu peran. Rincian struktur kolom tabel ini dapat dilihat pada Tabel 3.4.

Tabel 3.4. Struktur Tabel `user_role`

Nama Kolom	Tipe Data	Constraint
user_id	INTEGER	PRIMARY KEY, FOREIGN KEY (users.id)
role_id	INTEGER	PRIMARY KEY, FOREIGN KEY (roles.id)

Berikut merupakan penjelasan setiap atribut pada tabel `user_role`:

1. `user_id`: Kunci asing yang merujuk pada `id` pengguna dari tabel `users`. Kolom ini merupakan bagian dari kunci utama tabel.
2. `role_id`: Kunci asing yang merujuk pada `id` peran dari tabel `roles`. Kolom ini merupakan bagian dari kunci utama tabel.

B.4 Tabel `permissions`

Tabel `permissions` menyimpan daftar izin-izin spesifik yang dapat diberikan dalam sistem, seperti ”READ_USER” atau ”CREATE_USER”, sebagai bagian dari implementasi RBAC. Izin-izin ini akan diasosiasikan dengan peran tertentu melalui

tabel role_permission. Rincian struktur kolom tabel ini dapat dilihat pada Tabel 3.5.

Tabel 3.5. Struktur Tabel permissions

Nama Kolom	Tipe Data	Constraint
id	INTEGER	PRIMARY KEY, NOT NULL
name	VARCHAR	UNIQUE, NOT NULL

Berikut merupakan penjelasan setiap atribut pada tabel permissions:

1. id: Pengidentifikasi unik untuk setiap izin. Kolom ini berfungsi sebagai kunci utama tabel dan tidak boleh kosong.
2. name: Nama unik dari izin yang dimiliki sistem. Nilai pada kolom ini harus unik dan tidak boleh kosong.

B.5 Tabel role_permission

Tabel role_permission adalah tabel penghubung yang mereferensikan id dari tabel roles dan id dari tabel permissions, berfungsi untuk mendefinisikan relasi antara peran dan izin. Relasi ini memungkinkan satu peran untuk memiliki banyak izin. Rincian struktur kolom tabel ini dapat dilihat pada Tabel 3.6.

Tabel 3.6. Struktur Tabel role_permission

Nama Kolom	Tipe Data	Constraint
role_id	INTEGER	PRIMARY KEY, FOREIGN KEY (roles.id)
permission_id	INTEGER	PRIMARY KEY, FOREIGN KEY (permissions.id)

Berikut merupakan penjelasan setiap atribut pada tabel role_permission:

1. role_id: Kunci asing yang merujuk pada id peran dari tabel roles. Kolom ini merupakan bagian dari kunci utama tabel.
2. permission_id: Kunci asing yang merujuk pada id izin dari tabel permissions. Kolom ini merupakan bagian dari kunci utama tabel.

B.6 Tabel `events`

Tabel `events` berfungsi untuk menyimpan seluruh detail informasi mengenai setiap acara yang dibuat. Setiap acara dalam tabel ini mereferensikan `id` dari tabel `users` sebagai pembuat acara, dan pada gilirannya, tabel guests akan mereferensikan `id` dari tabel `events`. Rincian struktur kolom tabel ini dapat dilihat pada Tabel 3.7.

Tabel 3.7. Struktur Tabel `events`

Nama Kolom	Tipe Data	Constraint
id	INTEGER	PRIMARY KEY, NOT NULL
user_id	INTEGER	NOT NULL, FOREIGN KEY (users.id)
name	VARCHAR	NOT NULL
location	TEXT	-
description	TEXT	-
max_guests	INTEGER	-
event_date_time	TIMESTAMP	
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP

Berikut merupakan penjelasan setiap atribut pada tabel `events`:

1. `id`: Pengidentifikasi unik untuk setiap acara. Kolom ini berfungsi sebagai kunci utama tabel dan tidak boleh kosong.
2. `user_id`: Kunci asing yang merujuk pada `id` pengguna yang membuat acara. Kolom ini tidak boleh kosong.
3. `name`: Nama atau judul acara. Kolom ini tidak boleh kosong.
4. `location`: Lokasi atau tempat acara diselenggarakan.
5. `description`: Deskripsi detail mengenai acara.
6. `max_guests`: Kapasitas maksimum jumlah tamu yang diizinkan untuk acara ini.
7. `event_date_time`: Tanggal dan waktu pelaksanaan acara.
8. `created_at`: Tanggal dan waktu acara dibuat. Jika tidak diisi, nilai defaultnya adalah waktu saat ini.

9. `updated_at`: Tanggal dan waktu terakhir detail acara diperbarui. Jika tidak diisi, nilai defaultnya adalah waktu saat ini.

B.7 Tabel *guests*

Tabel `guests` berfungsi untuk menyimpan informasi tamu yang terkait dengan suatu acara, termasuk status kehadiran dan detail RSVP. Setiap data tamu mereferensikan `id` dari tabel `events`. Rincian struktur kolom tabel ini dapat dilihat pada Tabel 3.8.

Tabel 3.8. Struktur Tabel `guests`

Nama Kolom	Tipe Data	Constraint
id	INTEGER	PRIMARY KEY, NOT NULL
event_id	INTEGER	NOT NULL, FOREIGN KEY (events.id)
name	VARCHAR	NOT NULL
phone	VARCHAR	-
is_attending	BOOLEAN	-
attending_count	INTEGER	DEFAULT 1
created_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
updated_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
is_checked_in	BOOLEAN	DEFAULT FALSE
rsvp_code	VARCHAR	UNIQUE
qr_image	BYTEA	-
is_message_sent	BOOLEAN	DEFAULT FALSE
primary_guest	BIGINT	-
is_external	BOOLEAN	DEFAULT FALSE
is_rsvp	BOOLEAN	DEFAULT FALSE

Berikut merupakan penjelasan setiap atribut pada tabel `guests`:

1. `id`: Pengidentifikasi unik untuk setiap tamu. Kolom ini berfungsi sebagai kunci utama tabel dan tidak boleh kosong.
2. `event_id`: Kunci asing yang merujuk pada `id` acara yang mengundang tamu ini. Kolom ini tidak boleh kosong.
3. `name`: Nama lengkap tamu. Kolom ini tidak boleh kosong.

4. `phone`: Nomor telepon kontak tamu.

5. `is_attending`: Status boolean yang menunjukkan apakah tamu menyatakan akan hadir.

6. `attending_count`: Jumlah orang yang akan hadir di bawah nama tamu utama ini. Jika tidak diisi, nilai defaultnya adalah 1.

7. `created_at`: Tanggal dan waktu data tamu dibuat. Jika tidak diisi, nilai defaultnya adalah waktu saat ini.

8. `updated_at`: Tanggal dan waktu terakhir data tamu diperbarui. Jika tidak diisi, nilai defaultnya adalah waktu saat ini.

9. `is_checked_in`: Status boolean yang menunjukkan apakah tamu sudah melakukan *check-in* di acara. Jika tidak diisi, nilai defaultnya adalah *FALSE*.

10. `rsvp_code`: Kode unik yang diberikan kepada tamu untuk konfirmasi kehadiran (RSVP) dan verifikasi saat proses pemindaian. Nilai pada kolom ini harus unik.

11. `qr_image`: Data biner dari gambar kode QR yang terkait dengan `rsvp_code`, digunakan untuk proses pemindaian dan verifikasi.

12. `is_message_sent`: Status boolean yang menunjukkan apakah pesan atau undangan telah berhasil dikirimkan kepada tamu. Jika tidak diisi, nilai defaultnya adalah *FALSE*.

13. `primary_guest`: Kunci asing opsional yang merujuk pada id tamu utama, digunakan untuk mengelompokkan tamu sekunder di bawah satu entri tamu utama.

14. `is_external`: Status boolean yang menunjukkan apakah tamu diundang dari sumber eksternal (bukan dari daftar pengguna terdaftar). Jika tidak diisi, nilai defaultnya adalah *FALSE*.

15. `is_rsvp`: Status boolean yang menunjukkan apakah tamu telah melakukan proses RSVP (mengisi formulir konfirmasi kehadiran). Jika tidak diisi, nilai defaultnya adalah *FALSE*.

B.8 Tabel `sessions`

Tabel `sessions` berfungsi untuk menyimpan data sesi pengguna, khususnya untuk manajemen *refresh token* dan pelacakan sesi per perangkat dalam mekanisme autentikasi. Tabel ini mereferensikan `id` dari tabel `users`. Rincian struktur kolom tabel ini dapat dilihat pada Tabel 3.9.

Tabel 3.9. Struktur Tabel `sessions`

Nama Kolom	Tipe Data	Constraint
id	INTEGER	PRIMARY KEY, NOT NULL
user_id	INTEGER	FOREIGN KEY (users.id)
refresh_token	VARCHAR	NOT NULL
last_used_at	TIMESTAMP	DEFAULT CURRENT_TIMESTAMP
expired_at	TIMESTAMP	-
device_id	VARCHAR	-

Berikut merupakan penjelasan setiap atribut pada tabel `sessions`:

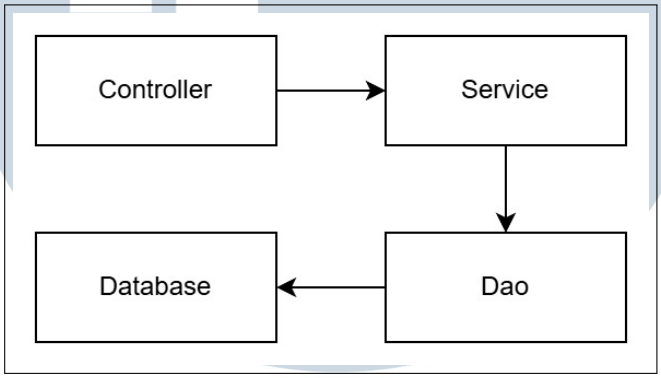
1. `id`: Pengidentifikasi unik untuk setiap sesi. Kolom ini berfungsi sebagai kunci utama tabel dan tidak boleh kosong.
2. `user_id`: Kunci asing yang merujuk pada `id` pengguna yang memiliki sesi ini.
3. `refresh_token`: Token unik yang digunakan untuk memperbarui token akses dan mempertahankan sesi pengguna. Kolom ini tidak boleh kosong.
4. `last_used_at`: Tanggal dan waktu terakhir sesi ini digunakan atau diperbarui. Jika tidak diisi, nilai defaultnya adalah waktu saat ini.
5. `expired_at`: Tanggal dan waktu masa berlaku refresh token akan berakhir.
6. `device_id`: Pengidentifikasi unik untuk perangkat tempat sesi ini aktif.

C Perancangan Komponen (Class Diagram)

Arsitektur backend sistem *e-invitation* ini dirancang dengan mengadopsi pola berlapis (*layered architecture*) untuk memastikan pemisahan tanggung jawab (*separation of concerns*) yang jelas antar komponen. Pola ini, seperti yang

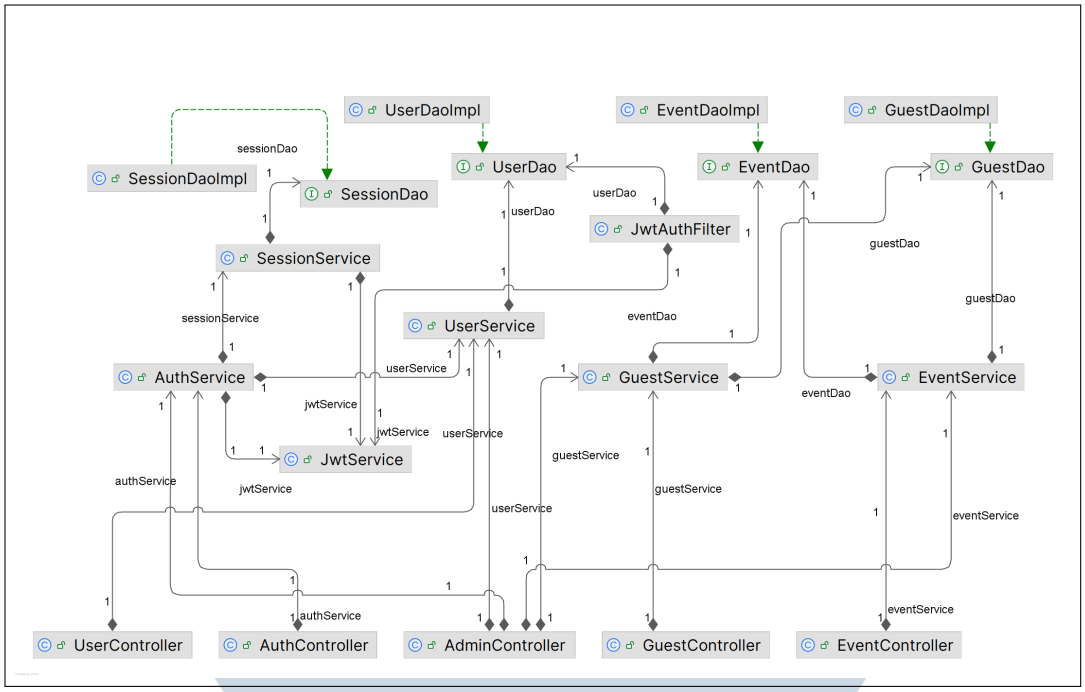
diilustrasikan secara konseptual pada Gambar 3.5, mengatur alur permintaan secara sistematis:

1. `Controller`: Lapisan terluar yang berfungsi sebagai titik masuk (*entry point*), menerima permintaan dari klien (misalnya, permintaan HTTP).
2. `Service`: Lapisan tengah yang berisi semua logika bisnis inti dari aplikasi.
3. `DAO` (*Data Access Object*): Lapisan yang bertanggung jawab secara eksklusif untuk berinteraksi dengan sumber data.



Gambar 3.5. Diagram Alur Komunikasi

Pola arsitektur berlapis tersebut kemudian diimplementasikan secara konkret melalui struktur kelas yang dapat dilihat pada Gambar 3.6. Diagram ini menunjukkan bagaimana setiap *class* saling berinteraksi dalam kerangka arsitektur yang telah ditetapkan, meskipun detail atribut dan *method* pada *class* tidak disertakan untuk menjaga kerahasiaan logika bisnis.



Gambar 3.6. Class Diagram E-invitation Minyma

Alur interaksi antar komponen dapat diilustrasikan melalui contoh proses autentikasi:

1. Lapisan `Controller`: Interaksi dimulai ketika `AuthController` menerima permintaan login dari pengguna.
2. Lapisan `Service`: `AuthController` kemudian meneruskan data permintaan ke `AuthService`. Di dalam lapisan servis, `AuthService` akan berkolaborasi dengan servis lain untuk mengeksekusi logika bisnis, seperti memanggil `UserService` untuk memvalidasi kredensial pengguna ke data yang ada, dan berkomunikasi dengan `JwtService` untuk membuat *JSON Web Token* (JWT) jika autentikasi berhasil.
3. Lapisan `DAO`: Jika `UserService` perlu mengambil data pengguna dari basis data, ia tidak akan mengaksesnya secara langsung, melainkan akan memanggil `UserDao`. `UserDao` inilah yang menjalankan kueri untuk mengambil atau memanipulasi data pengguna dari basis data.

Dengan struktur ini, setiap komponen memiliki tanggung jawab yang terisolasi dan jelas: `Controller` untuk antarmuka, `Service` untuk proses bisnis, dan `DAO` untuk

akses data. Hal ini membuat keseluruhan sistem menjadi lebih modular, mudah diuji (*testable*), dan mudah untuk dikelola atau dikembangkan di kemudian hari.

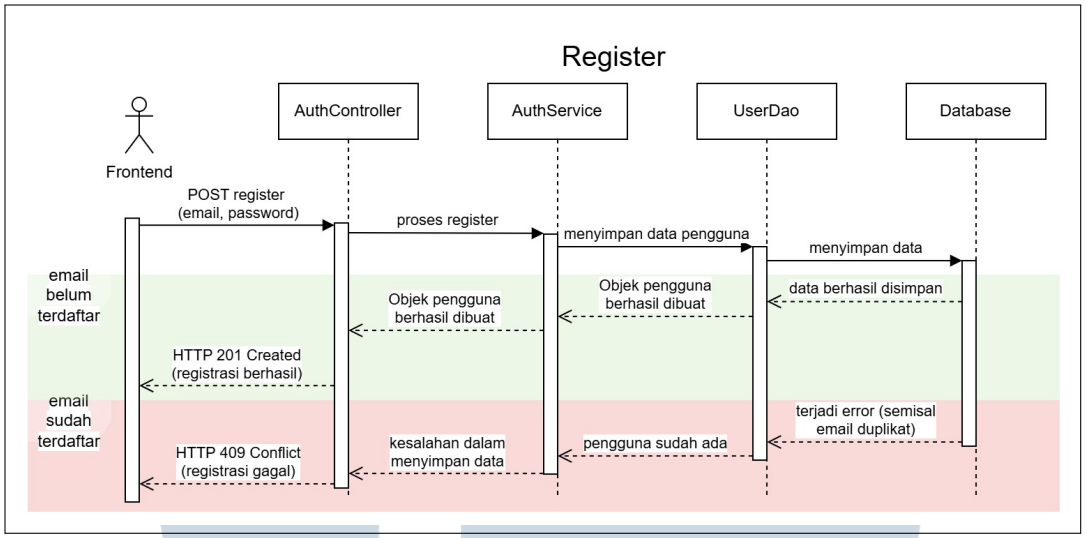
D Perancangan Proses (Sequence & Activity Diagram)

Pada bagian ini, akan diuraikan secara mendalam mengenai perancangan dan alur proses bisnis serta interaksi antar komponen sistem *backend*. Pemodelan ini dilakukan menggunakan *Sequence Diagram* untuk menggambarkan urutan komunikasi dan *Activity Diagram* untuk menunjukkan alur kontrol.

D.1 Alur Autentikasi dan Sesi

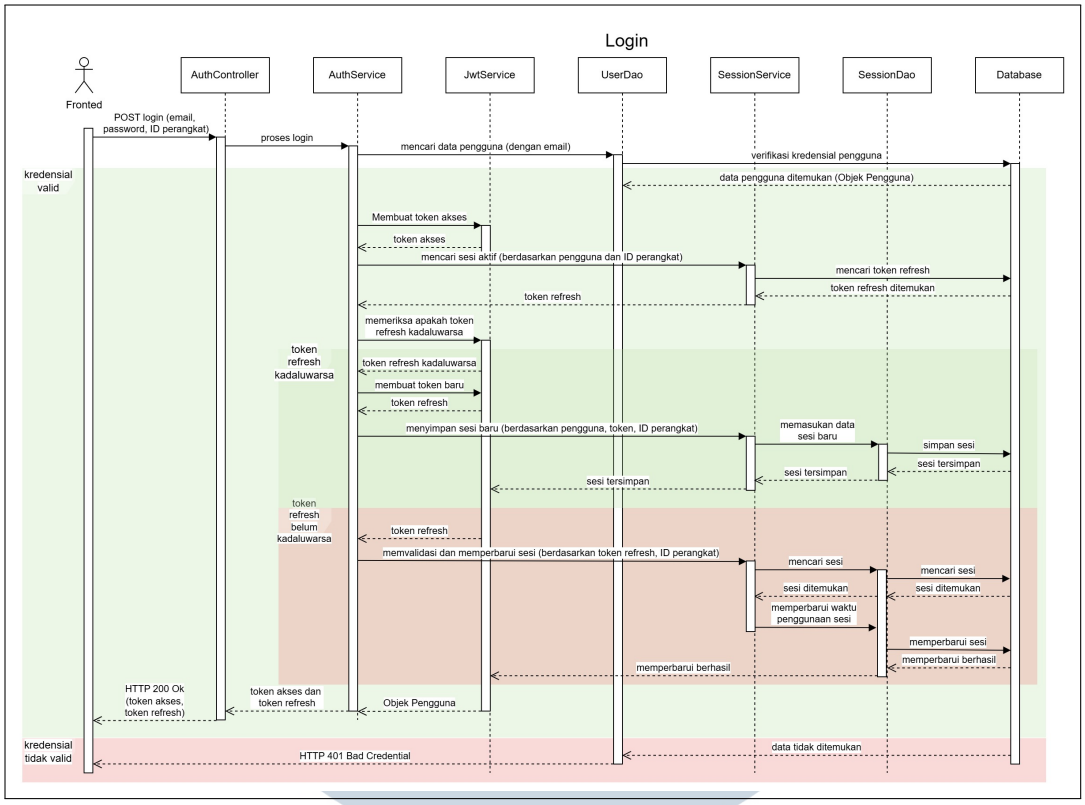
Arsitektur keamanan sistem dirancang untuk mengelola siklus akses pengguna secara lengkap, dimulai dari proses pendaftaran akun baru, autentikasi untuk memulai sesi, hingga pengguna mengakhiri sesi tersebut. Setiap tahapan diatur melalui proses dan komponen *backend* yang spesifik.

Langkah pertama bagi pengguna baru untuk dapat mengakses aplikasi adalah melalui proses registrasi. Interaksi awal pengguna terjadi melalui *frontend* aplikasi, yang kemudian akan meneruskan permintaan dan data yang relevan ke *backend* untuk diproses. Seperti yang diilustrasikan pada Gambar 3.7, pengguna akan mengirimkan data pendaftaran berupa `email` dan `password`. Permintaan akan diterima oleh *Controller* dan diteruskan ke lapisan *Service* untuk diproses diteruskan oleh *Backend*. Terdapat dua skenario utama yang mungkin terjadi: data pengguna berhasil disimpan atau terjadi kegagalan. Kegagalan umum disebabkan oleh upaya pendaftaran dengan alamat `email` yang sudah terdaftar, karena setiap `email` dalam basis data harus bersifat unik. Penting dicatat bahwa ketika `password` diterima oleh *Backend*, `password` tersebut akan segera di-*hashing* menggunakan algoritma `Bcrypt` sebelum disimpan ke basis data. Penggunaan `Bcrypt` ini bertujuan untuk mengamankan *password* sehingga `password` asli tidak akan pernah dapat diakses atau diketahui oleh pihak manapun, termasuk oleh sistem *Backend* atau administrator basis data. Basis data juga tidak akan menyimpan informasi apapun yang diberikan pengguna jika terjadi kesalahan atau validasi gagal.



Gambar 3.7. Sequence Diagram Register

Setelah pengguna berhasil terdaftar, proses autentikasi selanjutnya adalah login untuk mendapatkan akses ke fitur aplikasi. Seperti yang diilustrasikan pada Gambar 3.8, proses login diawali dengan *frontend* mengirimkan `email` dan `password` yang sebelumnya didapatkan dari pengguna dan juga identifikasi perangkat (`device_id`) ke `Backend`. Nilai dari `device_id` didapatkan dari pembuatan *String UUID* acak yang nantinya akan `Frontend` simpan di *cookie* atau *localStorage* setiap kali pengguna pertama kali mengakses website. Permintaan akan diterima oleh *Controller* dan diteruskan ke lapisan `AuthService` untuk diproses.



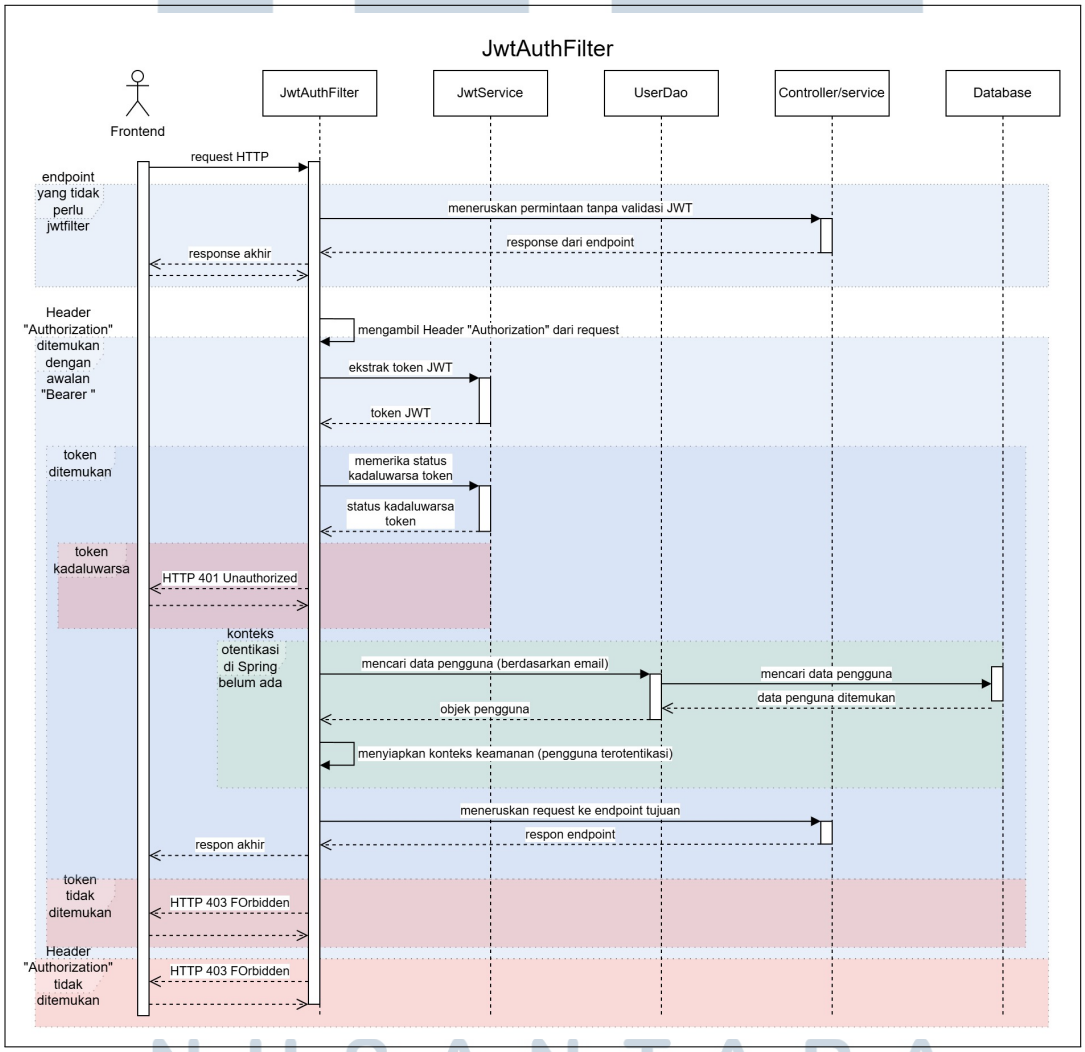
Gambar 3.8. Sequence Diagram Login

Pada lapisan `AuthService`, data pengguna akan dicari berdasarkan `email` yang diberikan. Setelah data pengguna ditemukan, kredensial yang diberikan akan diverifikasi. Jika kredensial tidak valid, respon *HTTP 401 Bad Credential* akan dikirimkan kepada `Frontend`, menandakan kegagalan login.

Apabila kredensial terbukti valid dan data pengguna ada, sistem akan melanjutkan proses dengan membuat dua token di `JwtService`, yaitu `token akses` dan `token refresh` yang nantinya `token refresh` akan disimpan di basis data beserta dengan `device_id` dan data pengguna, dalam hal ini `id`. `Token akses` berfungsi agar pengguna dapat mengakses fitur website yang perlu kredensial dan memilki waktu kadaluwarsa yang relatif pendek, dalam konteks ini 15 menit. Sedangkan `token refresh` berfungsi untuk membuat kembali `token akses` jika sudah kadaluwarsa agar pengguna dapat mengakses fitur website kembali dan memilki waktu kadaluwarsa yang lebih lama.

Selanjutnya `Backend` akan mengembalikan respon ke `Frontend` berupa *HTTP 200 Ok* beserta dengan `token akses` dan `token refresh`. Kedua token tersebut akan disimpan oleh `Frontend` di *Cookie* atau *localStorage* dan dipanggil setiap kali pengguna mengakses fitur website yang perlu data pengguna.

Setiap kali pengguna mengakses suatu endpoint maka dari sisi *server* perlu divalidasi apakah pengguna tersebut memiliki akses ke *endpoint* tersebut atau tidak. Seperti yang sebelumnya disebutkan bahwa aktor utama ada sistem ini adalah `Pengguna` dan `Admin`, maka apapun yang bisa diakses oleh `Pengguna` maka juga bisa diakses oleh `Admin`. Untuk menjaga hal tersebut maka dibuatlah `JwtAuthFilter` yang diilustrasikan pada gambar 3.9. Setiap kali pengguna mengakses maka dari sisi `Frontend` akan memberikan `token akses` yang sebelumnya didapat dari proses login. Token ini akan menjadi acuan akses setiap *endpoint* yang digunakan.

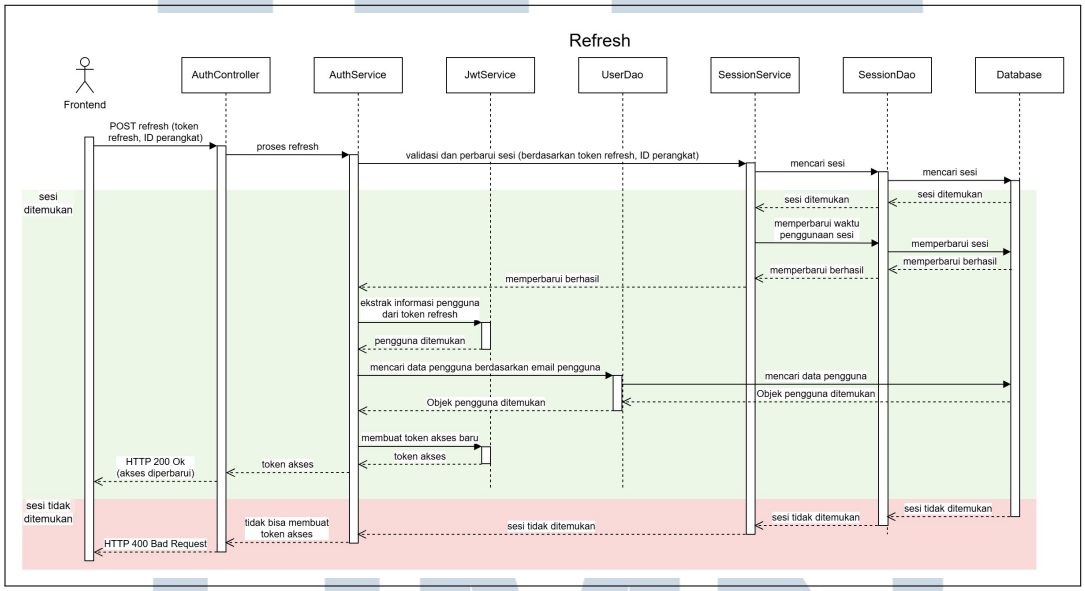


Gambar 3.9. Sequence Diagram JwtAuthFilter

Proses ekstrak dan pembuatan token menggunakan *JSON Web Token* (JWT). Karena dengan JWT, token dibuat dengan informasi pengguna dan dapat divalidasi oleh

basis data. Sehingga setiap aktor yang terlibat diletakkan di tempatnya masing-masing.

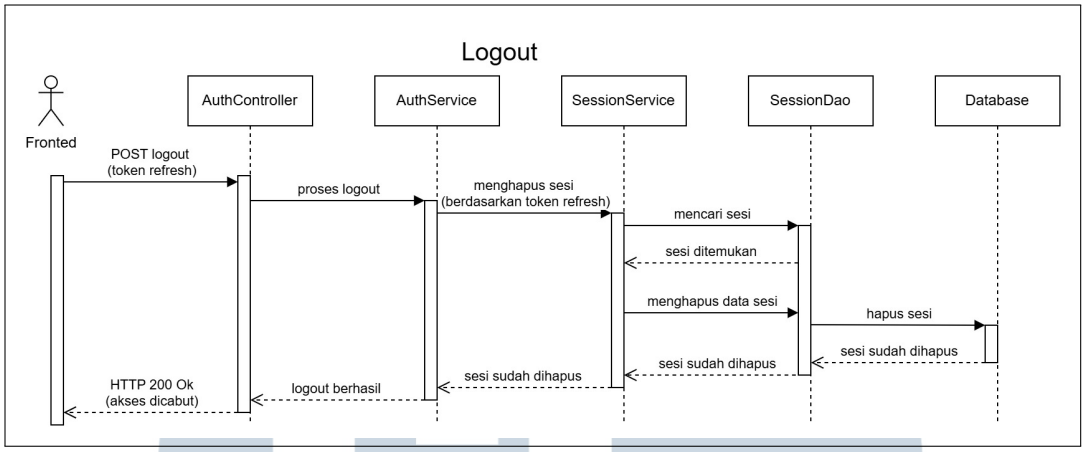
Ketika `token akses` yang dimiliki pengguna sudah masuk dalam masa kadaluwarsa dan `token refresh` masih bisa digunakan, maka akan terjadi mekanisme memperbarui token. Proses ini dapat dilihat pada Gambar 3.10. `Token refresh` akan dikirim oleh `Frontend` dan divalidasi di *Backend*. Proses ini akan melakukan pengambilan informasi dari token yang selanjutnya akan membuat `token akses` baru dan disimpan di *cookie* atau *localStorage* agar pengguna tetap bisa mengakses endpoint tertentu.



Gambar 3.10. Sequence Diagram Refresh

Jika `token refresh` yang diterima oleh `Backend` ternyata sudah masuk ke waktu kadaluwarsa, maka pengguna sudah tidak bisa mengakses endpoint tersebut dan mengharuskan untuk login ulang untuk mendapatkan `token akses` dan `token refresh` baru.

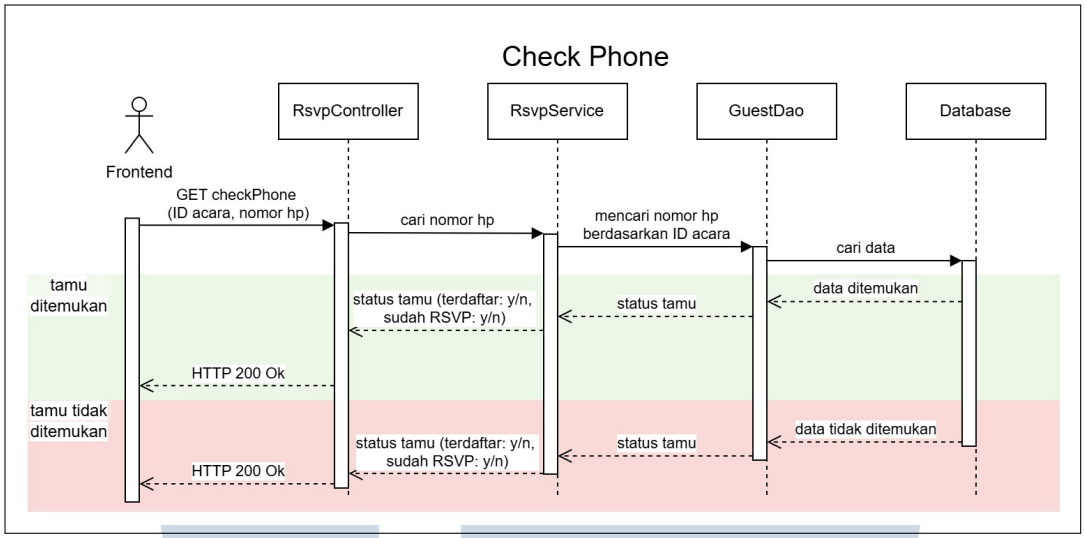
Setelah pengguna selesai untuk mengakses website, maka proses yang bisa dilakukan adalah logout. Seperti pada Gambar 3.11, permintaan logout diteruskan dari pengguna ke `Frontend` dengan mengirimkan `token refresh` yang dimiliki pengguna. Token tersebut akan divalidasi oleh `Backend` dan dihapus di basis data. Sehingga kedepannya walaupun dari `Frontend` ada `token refresh` tapi di basis data sudah tidak ada, maka pengguna tersebut tetap tidak memilik akses dan harus login ulang.



Gambar 3.11. Sequence Diagram Logout

D.2 Alur Manajemen Acara dan Tamu

Bagian ini akan menjelaskan proses manajemen (CRUD) dari layanan manajemen acara dan tamu. Dimulai dari bagaimana pembuatan acara, pembuatan tamu dan juga alur bagaimana tamu bisa mengonfirmasi kedatangan (RSVP) ke acara tertentu. Pada mekanisme RSVP, tamu harus divalidasi terlebih dahulu apakah akan datang, sudah terdaftar, dan sudah mengisi konfirmasi pada acara tertentu. Proses ini dapat dilihat pada Gambar 3.12, tamu akan mengisi nomor telepon pada formulir undangan acara. Selanjutnya, `Frontend` akan meneruskan ke `Backend` untuk memvalidasi apakah nomor tersebut terdaftar dan apakah sudah pernah mengisi formulir undangan.



Gambar 3.12. Sequence Diagram Check Phone

Proses selanjutnya ketika data apakah tamu sudah terdaftar (`isRegistered`), sudah mengisi form (`isRsvp`), dan ingin datang atau tidak (`isAttending`) sudah terkumpul adalah membagi logika bisnis yang ingin dijalankan. Seperti pada Gambar 3.13, terdapat beberapa kasus dan sub-kasus yang dapat terjadi, antara lain:

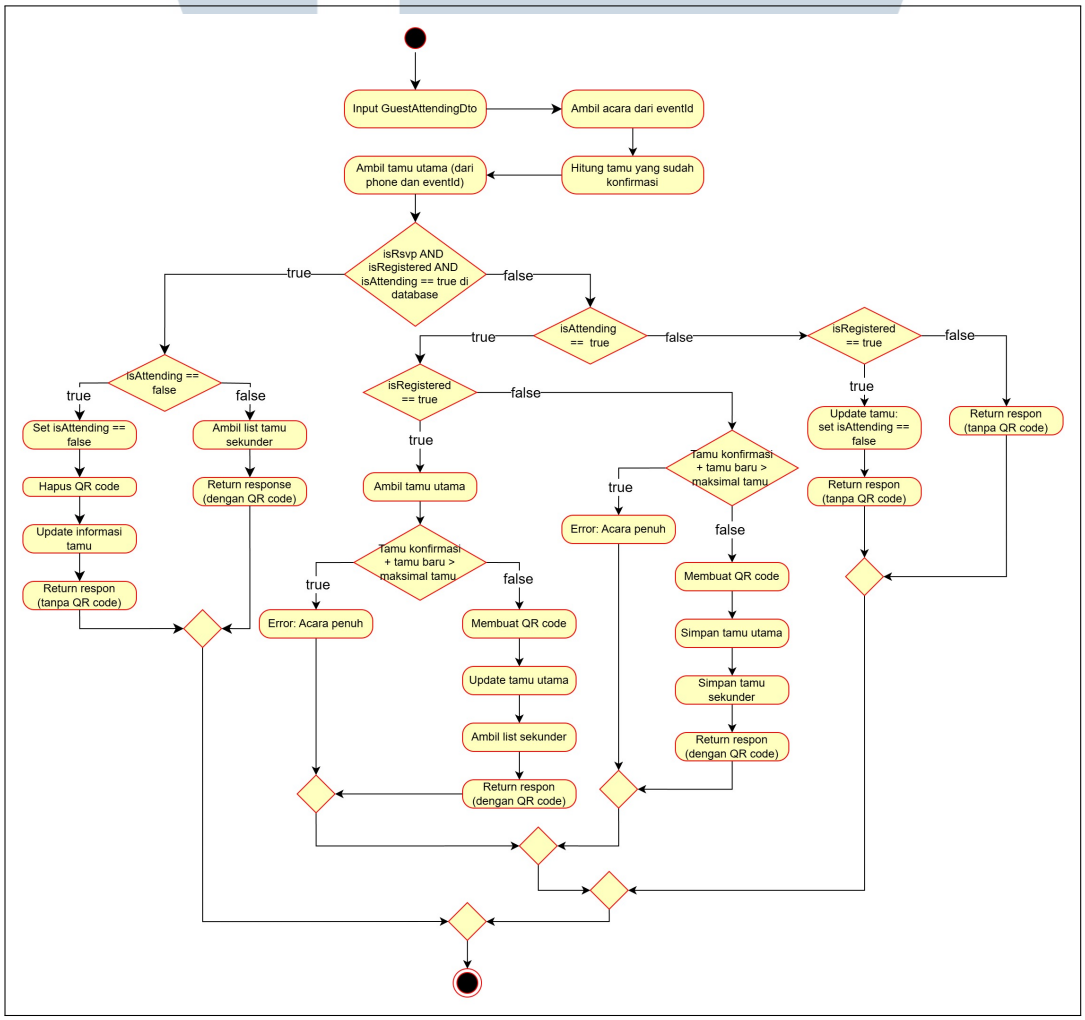
1. Tamu terdaftar dan sudah melakukan RSVP sebelumnya.
   (a) Jika tamu ingin membatalkan kehadiran (input `isAttending` bernilai *False*), maka sistem akan memperbarui status kehadiran tamu, menghapus kode RSVP yang sudah ada, dan mengembalikan respon tanpa kode QR.
   (b) Jika tamu yang sebelumnya batal hadir ingin mengonfirmasi ulang kehadirannya atau ingin menampilkan kode QR yang sudah didapatkan sebelumnya, sistem akan mengambil data tamu sekunder yang terkait dan mengembalikan respon dengan kode QR yang sudah ada sebelumnya.
2. Tamu akan datang dan namun baru pertama kali melakukan RSVP.
   (a) Jika tamu sudah terdaftar, sistem akan memeriksa kapasitas maksimal acara. Jika masih ada tempat, sistem akan membuat kode QR, memperbarui data tamu tersebut di basis data dan mengembalikan respon dengan kode QR. Jika maksimal kapasitas acara sudah terpenuhi, tamu tidak dapat mengisi formulir RSVP.

(b) Jika tamu belum terdaftar, sistem akan memeriksa kapasitas maksimal acara. Jika masih ada tempat, sistem akan membuat kode QR, menambahkan data tamu di basis data dan mengembalikan respon dengan kode QR. Jika maksimal kapasitas acara sudah terpenuhi, tamu tidak dapat mengisi formulir RSVP.

3. Tamu tidak akan datang dan namun baru pertama kali melakukan RSVP.

(a) Jika tamu sudah terdaftar, sistem akan memperbarui data tamu di basis data dan mengembalikan respon tanpa kode QR.

(b) Jika tamu belum terdaftar, sistem akan mengembalikan respon tanpa kode QR.



Gambar 3.13. Activity Diagram Sistem RSVP

3.3.3 Pembuatan dan Implementasi Application Programming Interface (API)

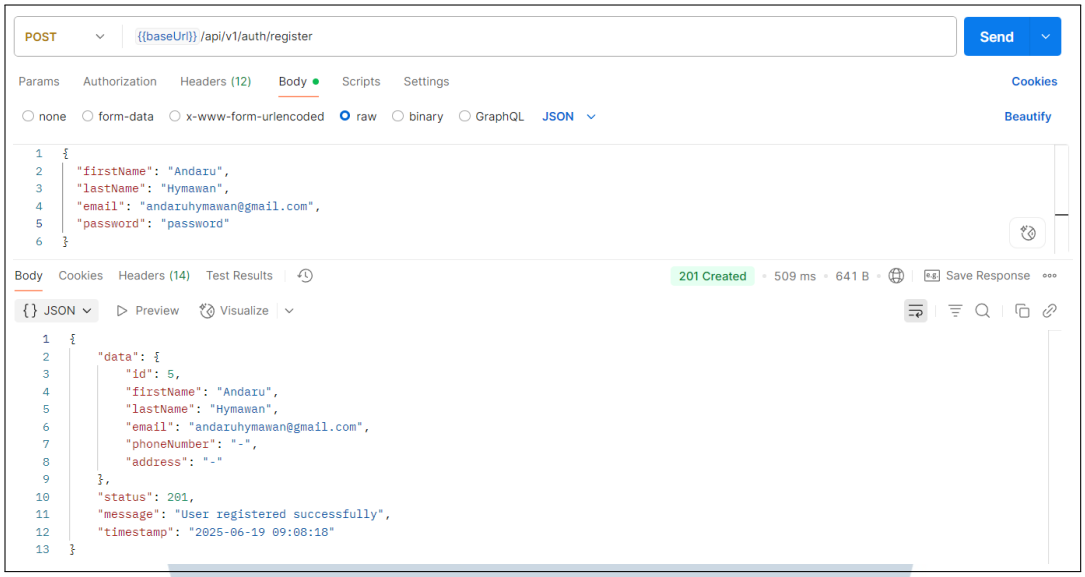
A API Autentikasi

Bagian ini akan menjelaskan implementasi API yang terkait dengan modul autentikasi sistem. Modul autentikasi berperan krusial dalam mengamankan akses pengguna ke aplikasi serta mengelola siklus hidup sesi mereka. API autentikasi utama yang diimplementasikan meliputi registrasi akun baru, login, pembaruan token, dan logout. Rincian mengenai *endpoint* dan metode HTTP untuk setiap API autentikasi dapat dilihat pada Tabel 3.10.

Tabel 3.10. Rincian Endpoint Autentikasi

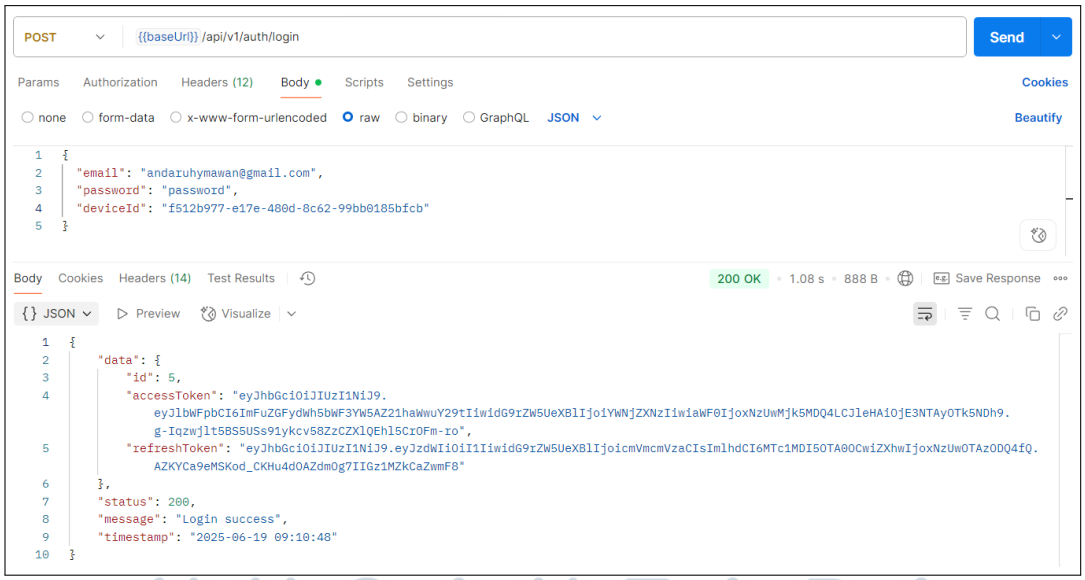
Nama Fitur	**Endpoint & Metode HTTP**	**Deskripsi Singkat**
Register	POST /api/v1/auth/register	Membuat akun pengguna baru
Login	POST /api/v1/auth/login	Mengautentikasi kredensial dan mengembalikan `accessToken` dan `refreshToken`.
Refresh Token	POST /api/v1/auth/refresh	Membuat `accessToken` baru dari `refreshToken` yang ada.
Logout	POST /api/v1/auth/logout	Menghapus sesi aktif berdasarkan `refreshToken`.

API *Registrasi* memfasilitasi pembuatan akun baru bagi pengguna dengan menerima data seperti `firstName`, `lastName`, `email`, dan `password`. Sistem akan memverifikasi keunikan `email`, dan berdasarkan hasilnya, akun dapat berhasil dibuat atau gagal jika email sudah terdaftar. Visualisasi proses ini dapat dilihat pada Gambar 3.14.



Gambar 3.14. Tampilan API Register di Postman

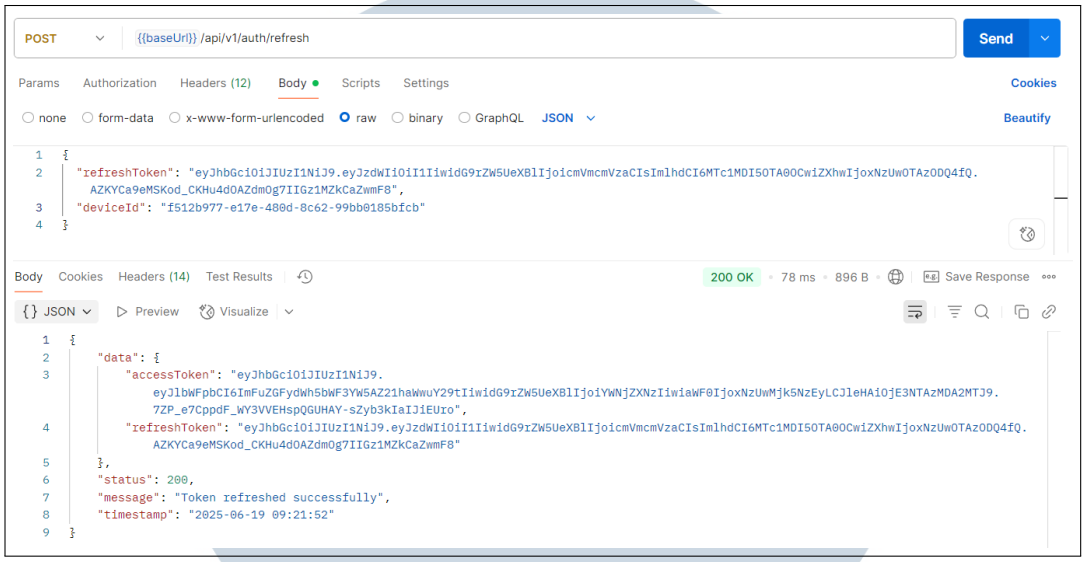
API *Login* berfungsi untuk mengotentikasi pengguna yang sudah terdaftar dengan menerima `email`, `password`, dan `deviceId`. Jika kredensial valid, sistem akan memberikan `token akses` dan `refresh token` untuk memulai sesi pengguna. Visualisasi proses ini dapat dilihat pada Gambar 3.15.



Gambar 3.15. Tampilan API Login di Postman

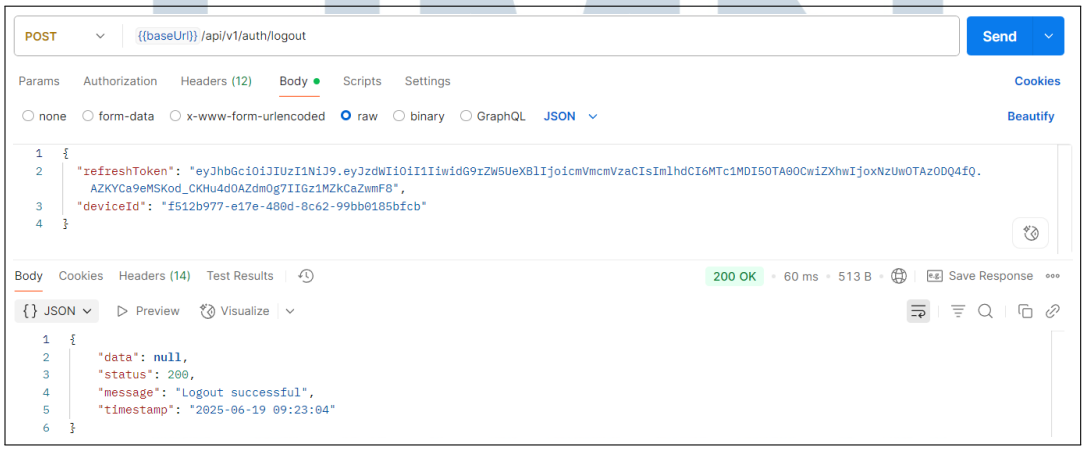
API *Refresh Token* memungkinkan pengguna mendapatkan `token akses` baru tanpa perlu login ulang, dengan mengirimkan `refreshToken` dan `deviceId` yang

valid. Jika proses berhasil, sistem akan memberikan `accessToken` yang diperbarui. Visualisasi proses ini dapat dilihat pada Gambar 3.16.



Gambar 3.16. Tampilan API Refresh di Postman

API *Logout* berfungsi untuk mengakhiri sesi pengguna yang sedang aktif dengan membatalkan `token refresh` yang dimiliki. Proses ini akan menghapus sesi pengguna dari sistem sehingga akses dicabut dan pengguna perlu login ulang untuk berinteraksi lebih lanjut. Visualisasi proses ini dapat dilihat pada Gambar 3.17.



Gambar 3.17. Tampilan API Logout di Postman

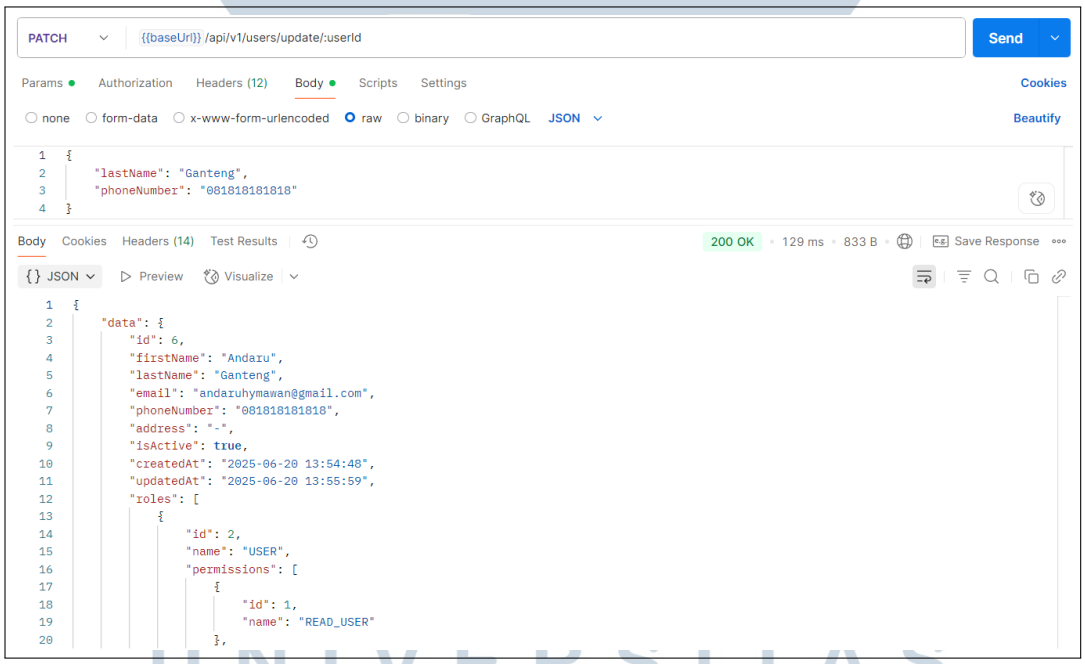
B API Manajemen Pengguna

Bagian ini akan menjelaskan implementasi API yang terkait dengan pengelolaan data pengguna dalam sistem. Fokus utama API ini adalah untuk memperbarui data yang dimiliki pengguna. Rincian *endpoint* dapat dilihat melalui Tabel 3.11.

Tabel 3.11. Rincian Endpoint User

Nama Fitur	Endpoint & Metode HTTP	Deskripsi Singkat
Update User	PATCH /api/v1/users/update/{userId}	Memperbarui data pengguna

API *Update User* memungkinkan `pengguna` dapat mengubah `firstName`, `lastName`, `phoneNumber`, dan `address` yang dimilikinya. Visualisasi dapat dilihat pada Gambar 3.18.



Gambar 3.18. Tampilan API Update User di Postman

C API Manajemen Acara

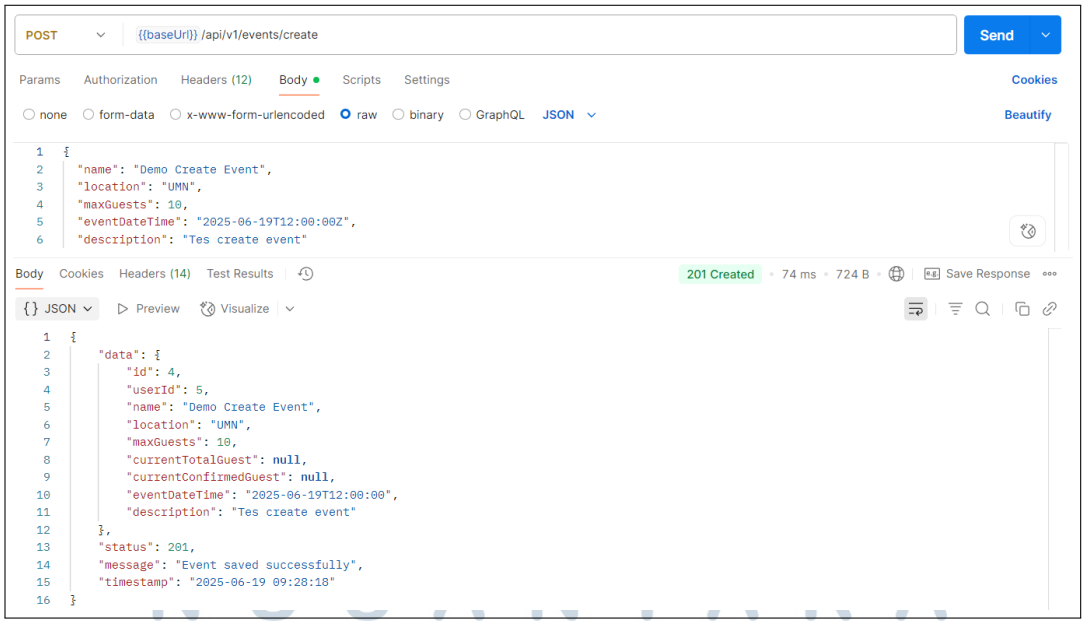
Bagian ini akan menjelaskan implementasi API yang terkait dengan pengelolaan data acara (*event*) dalam sistem. API Manajemen Acara berfungsi untuk memfasilitasi pengguna dalam membuat, mengubah, dan mencari informasi

mengenai acara yang diselenggarakan, memastikan bahwa data acara selalu akurat dan tersedia. Rincian *endpoint* dapat dilihat melalui Tabel 3.12.

Tabel 3.12. Rincian Endpoint Acara

Nama Fitur	**Endpoint & Metode HTTP**	**Deskripsi Singkat**
Create Event	POST /api/v1/events/create	Membuat acara baru
Update Event	PATCH /api/v1/events/update/{eventId}	Memperbarui data acara
Get Event	GET /api/v1/events/find/{eventId}	Membaca data detail acara
Get All Events	GET /api/v1/events/find-all	Membaca data semua acara

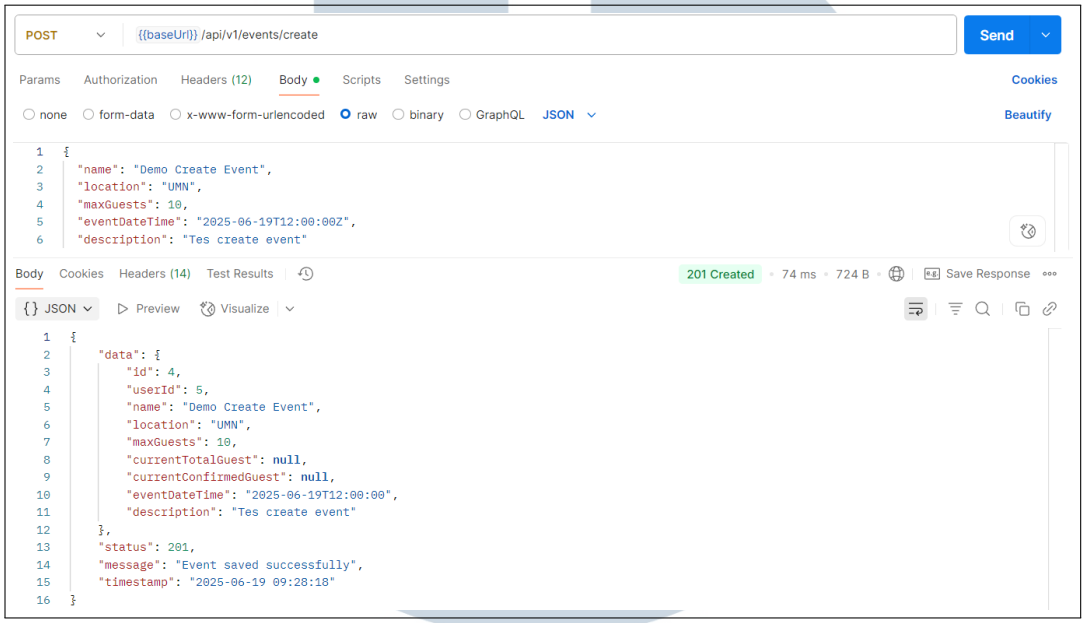
API *Create Event* memungkinkan pengguna untuk membuat acara baru dalam sistem dengan mengirimkan detail acara seperti name, location, maxGuest, `eventDateTime`, dan `description`. Jika pembuatan acara berhasil, sistem akan mengonfirmasi data acara yang telah dibuat. Visualisasi proses ini dapat dilihat pada Gambar 3.19.



Gambar 3.19. Tampilan API Create Event di Postman

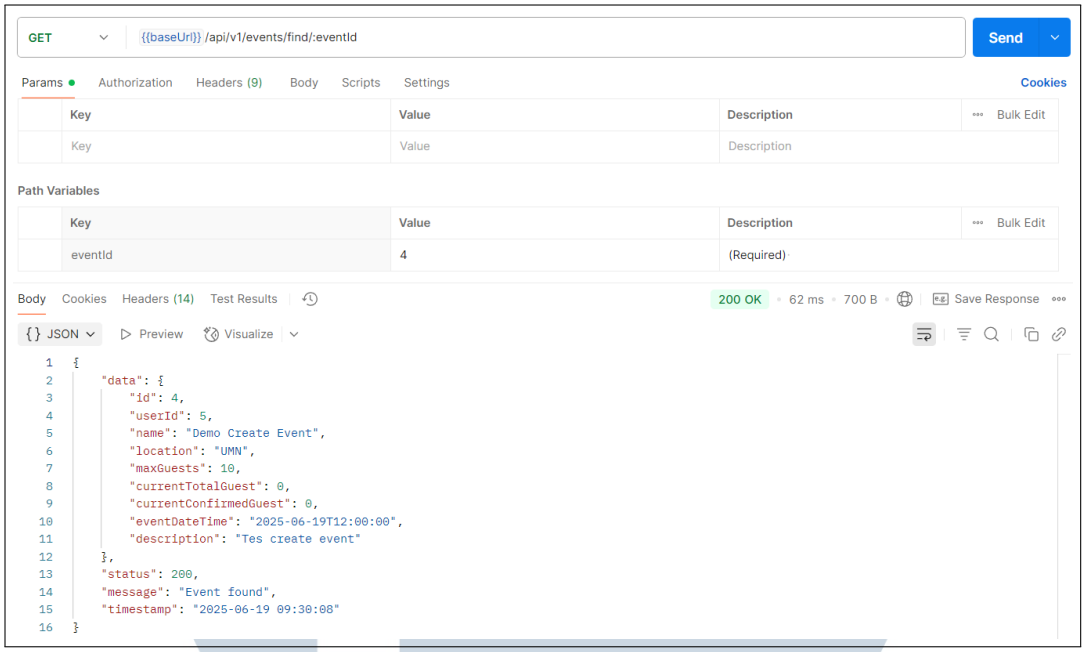
API *Update Event* berfungsi untuk memperbarui informasi acara yang sudah ada

di dalam sistem. Pengguna dapat mengirimkan perubahan pada detail acara seperti `description`, `location`, atau `eventDateTime`. Jika pembaruan berhasil, sistem akan mengkonfirmasi perubahan dan menampilkan detail acara yang telah diperbarui. Visualisasi proses ini dapat dilihat pada Gambar 3.20.



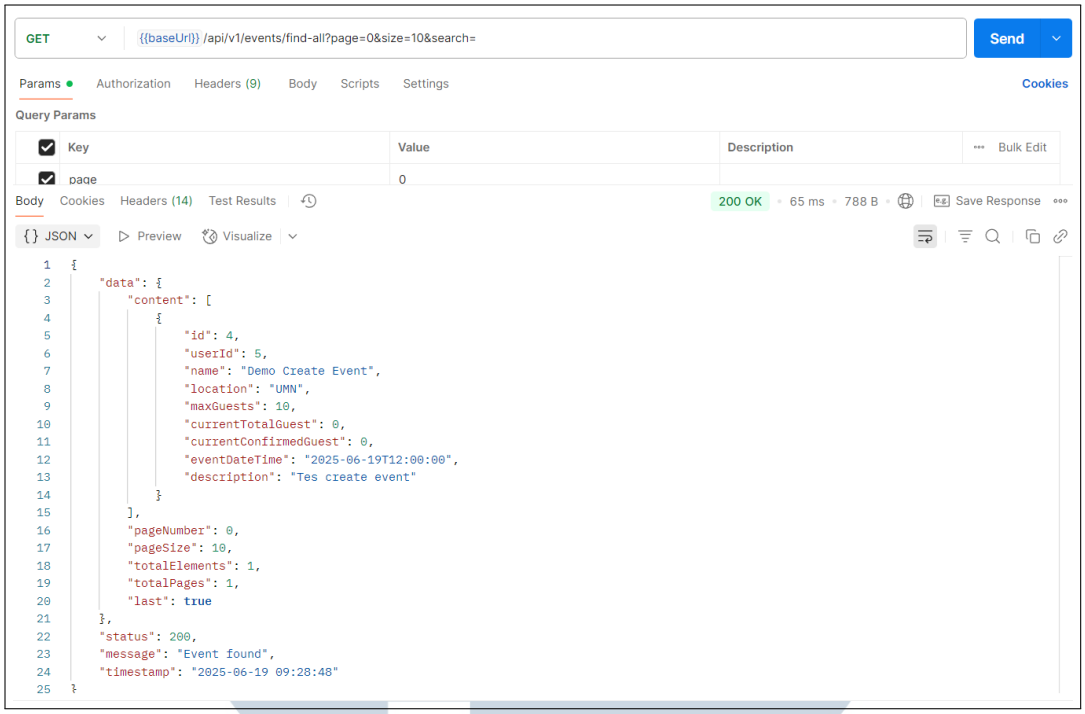
Gambar 3.20. Tampilan API Update Event di Postman

API *Find Event* berfungsi untuk mengambil informasi detail mengenai sebuah acara berdasarkan `id` uniknya. Dengan menyediakan `id` acara, pengguna dapat memperoleh semua data terkait acara tersebut dari sistem. Jika acara ditemukan, detail acara akan ditampilkan. Visualisasi proses ini dapat dilihat pada Gambar 3.21.



Gambar 3.21. Tampilan API Find Event di Postman

API *Find All Events* memungkinkan pengguna untuk mencari dan mengambil daftar semua acara yang tersedia dalam sistem, dengan dukungan untuk paginasi. Pengguna dapat menentukan parameter seperti nomor halaman dan ukuran data per halaman untuk menelusuri daftar acara. Hasilnya adalah daftar acara yang ditemukan beserta informasi paginasi. Visualisasi proses ini dapat dilihat pada Gambar 3.22.



Gambar 3.22. Tampilan API Find All Events di Postman

D API Manajemen Tamu

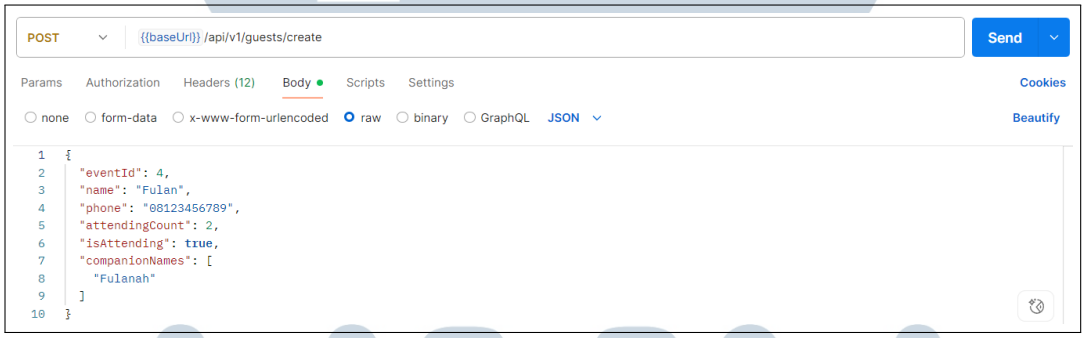
Bagian ini akan menjelaskan implementasi API yang terkait dengan pengelolaan data tamu untuk setiap acara dalam sistem. API Manajemen Tamu memungkinkan pengguna untuk menambah, mencari, dan menghapus data tamu, memastikan daftar tamu acara selalu akurat dan terorganisir. Rincian *endpoint* dapat dilihat melalui Tabel 3.13.

Tabel 3.13. Rincian Endpoint Manajemen Tamu

Nama Fitur	Endpoint & Metode HTTP	Deskripsi Singkat
Create Guest	POST /api/v1/guests/create	Membuat tamu baru
Create Multiple Guests	POST /api/v1/guests/create-batch/{eventId}	Membuat banyak tamu baru
Update Guest	PATCH /api/v1/guests/update/{guestId}	Memperbarui data tamu

Nama Fitur	Endpoint & Metode HTTP	Deskripsi Singkat
Get Guest	GET /api/v1/guests/find/{guestId}	Membaca data detail tamu
Get All Guests	GET /api/v1/guests/find-all/{eventId}	Membaca data semua tamu dalam suatu acara
Delete Guest	DELETE /api/v1/guests/delete/{guestId}	Menghapus tamu tertentu

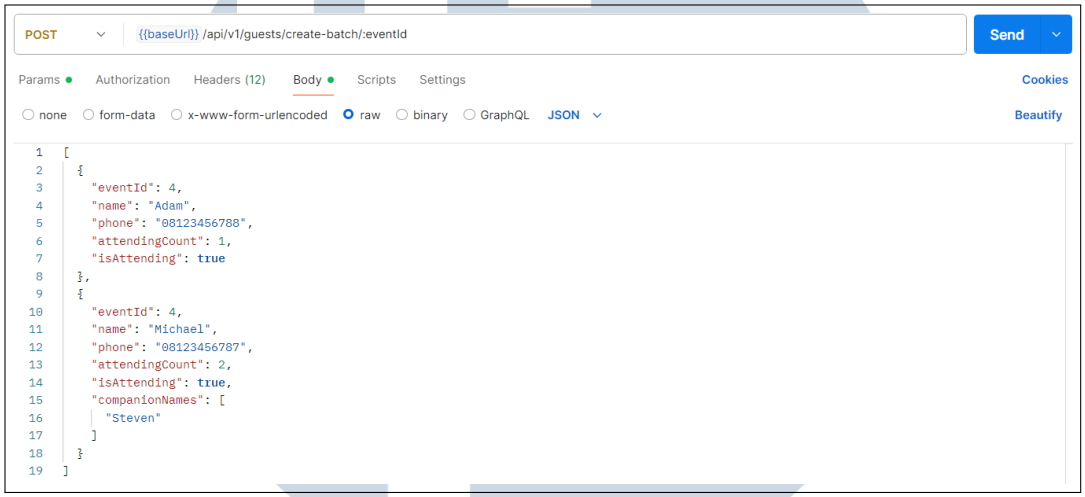
API *Create Guest* berfungsi untuk menambahkan satu data tamu baru ke dalam sistem untuk sebuah acara tertentu. Pengguna akan mengirimkan detail tamu seperti `name`, `phone`, dan `isAttendning`. Setelah berhasil dibuat, sistem akan mengonfirmasi data tamu yang telah tersimpan. Visualisasi permintaan dapat dilihat pada Gambar 3.23, dan responnya pada Gambar 3.24.



Gambar 3.23. Tampilan Permintaan API Create Guest di Postman

Gambar 3.24. Tampilan Respon API Create Guest di Postman

API *Create Multiple Guest* memungkinkan penambahan beberapa data tamu sekaligus untuk suatu acara. Pengguna dapat mengirimkan daftar tamu dalam satu permintaan untuk efisiensi. Jika proses berhasil, sistem akan mengonfirmasi data tamu-tamu yang telah ditambahkan. Visualisasi permintaan dapat dilihat pada Gambar 3.25, dan responnya pada Gambar 3.26.

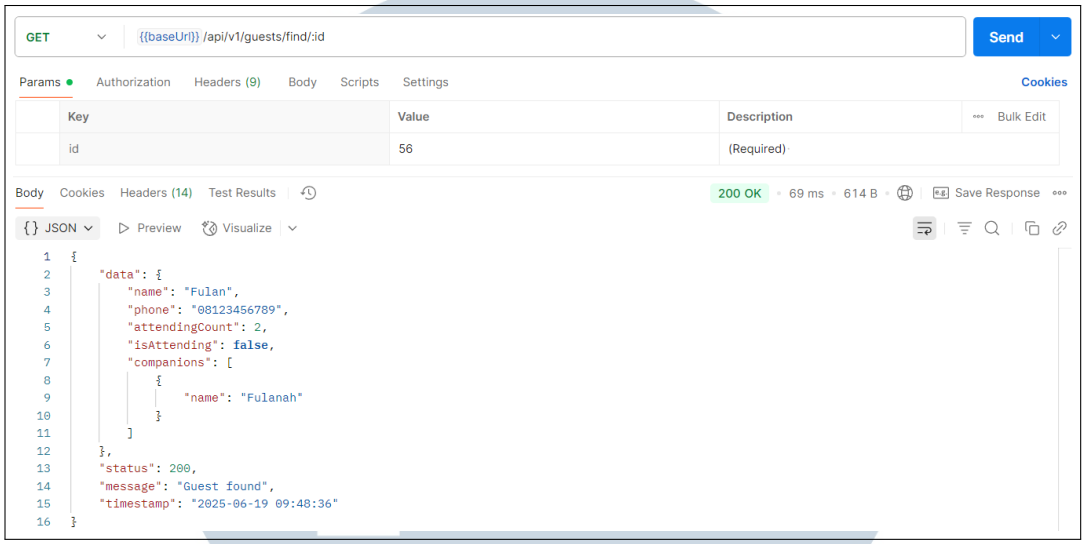


Gambar 3.25. Tampilan Permintaan API Create Multiple Guest di Postman

Gambar 3.26. Tampilan Respon API Create Multiple Guest di Postman

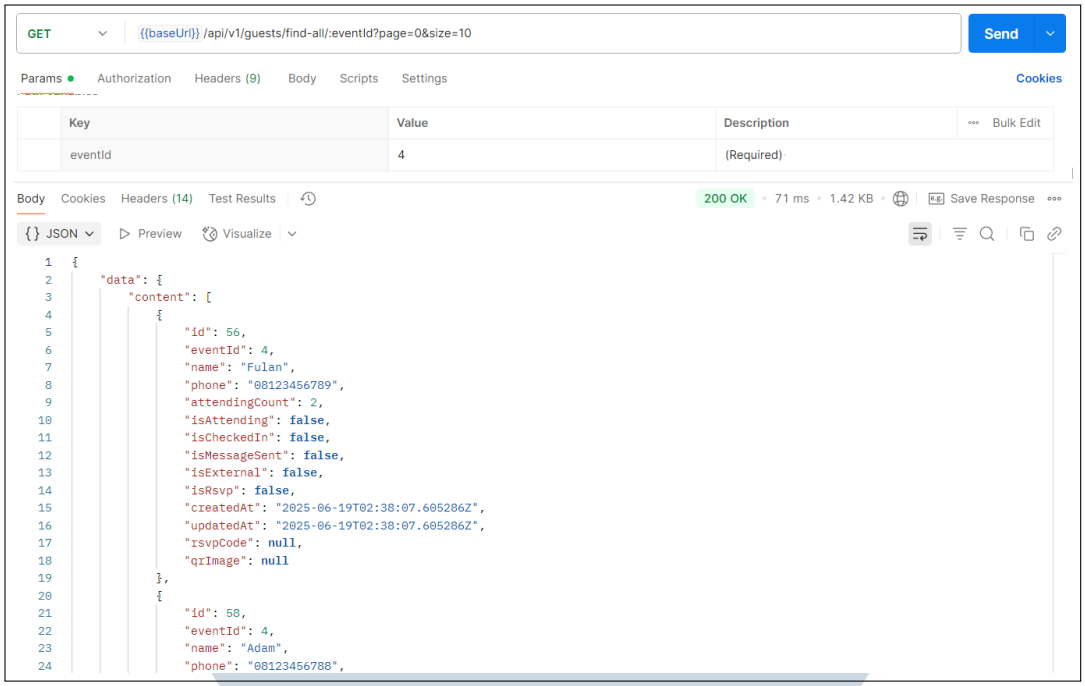
API *Find Guest* berfungsi untuk mengambil informasi detail mengenai satu tamu spesifik berdasarkan `id` uniknya. Dengan menyediakan `id` tamu, pengguna dapat

memperoleh semua data terkait tamu tersebut dari sistem. Jika tamu ditemukan, detailnya akan ditampilkan. Visualisasi proses ini dapat dilihat pada Gambar 3.27.



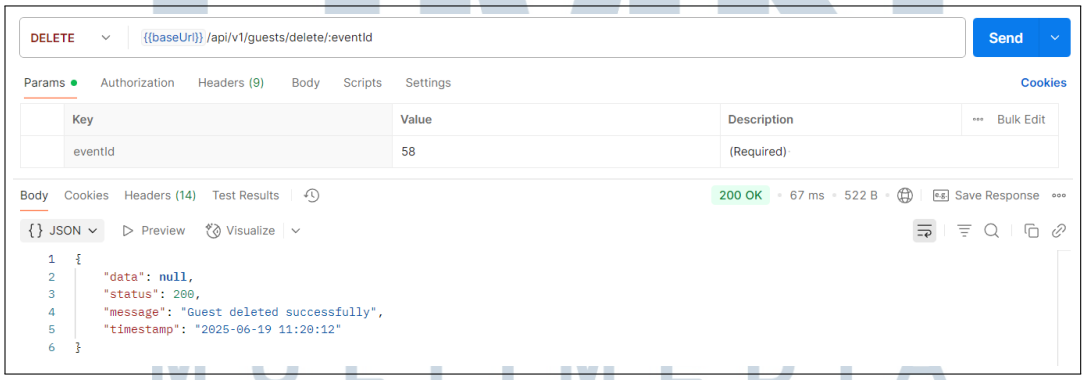
Gambar 3.27. Tampilan API Find Guest di Postman

API *Find All Guests* memungkinkan pengguna untuk mencari dan mengambil daftar semua tamu yang terdaftar untuk acara tertentu, dengan dukungan paginasi. Pengguna dapat menentukan `id` acara serta parameter halaman dan ukuran data. Hasilnya adalah daftar tamu yang ditemukan beserta informasi paginasi. Visualisasi proses ini dapat dilihat pada Gambar 3.28.



Gambar 3.28. Tampilan API Find All Guests di Postman

API *Delete Guest* berfungsi untuk menghapus data tamu spesifik dari sistem berdasarkan `id` tamu. Proses ini memastikan data tamu yang tidak lagi diperlukan dapat dikeluarkan dari daftar. Jika penghapusan berhasil, sistem akan mengonfirmasi bahwa data tamu telah dihapus. Visualisasi proses ini dapat dilihat pada Gambar 3.29.



Gambar 3.29. Tampilan API Delete Guests di Postman

E API Manajemen Global

Bagian ini menjelaskan implementasi API yang khusus dirancang untuk hak akses global, dalam hal ini aktor Admin. API Manajemen Global memberikan wewenang penuh kepada administrator untuk mengelola seluruh data sistem. Administrator dapat melakukan operasi apapun pada semua data acara, dan juga dapat melihat daftar lengkap semua tamu untuk setiap acara, terlepas dari pembuatnya. Rincian mengenai *endpoint* untuk setiap API Manajemen Global dapat dilihat pada Tabel 3.14.

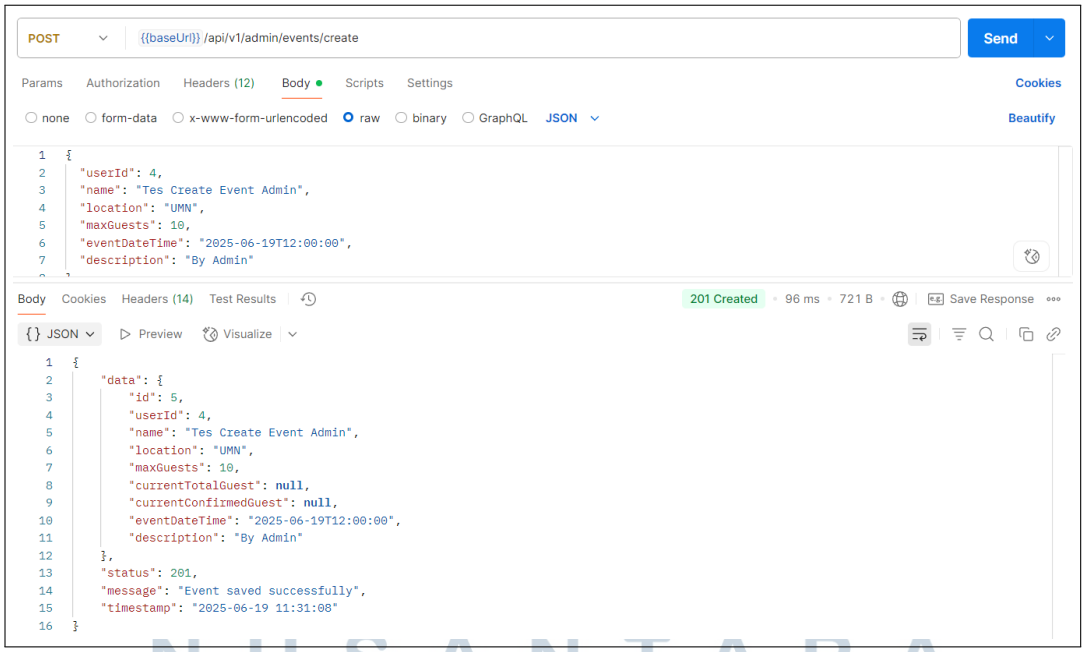
Tabel 3.14. Rincian Endpoint Manajemen Global

Nama Fitur	**Endpoint & Metode HTTP**	**Deskripsi Singkat**
Register User	POST /api/v1/admin/users/create	Membuat akun pengguna baru oleh admin
Find User	GET /api/v1/admin/users/find/userId	Membaca data detail pengguna
Find All Users	GET /api/v1/admin/users/find-all	Membaca data semua pengguna
Update User	PATCH /api/v1/admin/users/update/userId	Memperbarui data pengguna lain
Delete User	DELETE /api/v1/admin/users/delete/userId	Menghapus akun pengguna tertentu
Create Event	POST /api/v1/admin/create	Membuat acara baru
Update Event	PATCH /api/v1/admin/events/update/{eventId}	Memperbarui data acara
Find Event	GET /api/v1/admin/events/find/{eventId}	Membaca data detail acara
Find All Events	GET /api/v1/admin/events/find-all	Membaca data semua acara
Delete Event	DELETE /api/v1/admin/events/delete/{userId}	Menghapus acara tertentu
Find All Guests (admin)	GET /api/v1/admin/guests/find-all/{eventId}	Membaca data semua tamu dalam suatu acara

Implementasi dari API Manajemen Global ini memiliki alur dan respon yang serupa dengan pemaparan API di Sub-bab C API Manajemen Acara dan API Manajemen Tamu, namun dengan penambahan lingkup akses yang lebih luas untuk administrator. Untuk manajemen tamu, API Manajemen Global memungkinkan administrator untuk melihat daftar tamu yang dimiliki suatu acara tanpa bisa menambah, memperbarui, maupun menghapus data tamu melalui *endpoint-endpoint* ini.

Untuk dapat mengakses API Manajemen Global, pengguna harus login menggunakan akun yang memiliki `roles` sebagai `Admin`, yakni hanya pihak internal yang bisa melakukannya. Setelah berhasil login, administrator dapat mengakses *endpoint-endpoint* yang telah disebutkan.

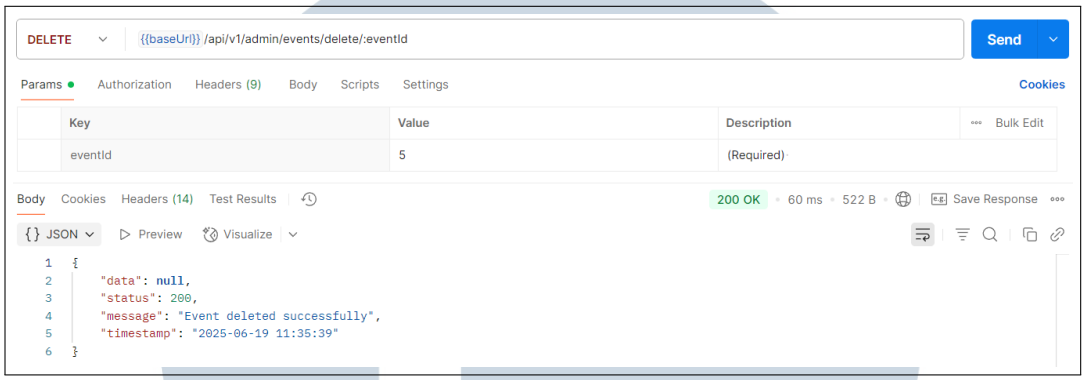
API *Create Event* memungkinkan Admin untuk membuat acara baru milik suatu pengguna. Detail yang perlu diberikan mengenai acara masih sama seperti yang bisa aktor Pengguna lakukan, bedanya adalah Admin perlu memberikan `userId` milik suatu Pengguna agar kepemilikan acara atas nama Pengguna tersebut. Visualisasi proses ini dapat dilihat pada Gambar 3.30



Gambar 3.30. Tampilan API Create Event di Postman

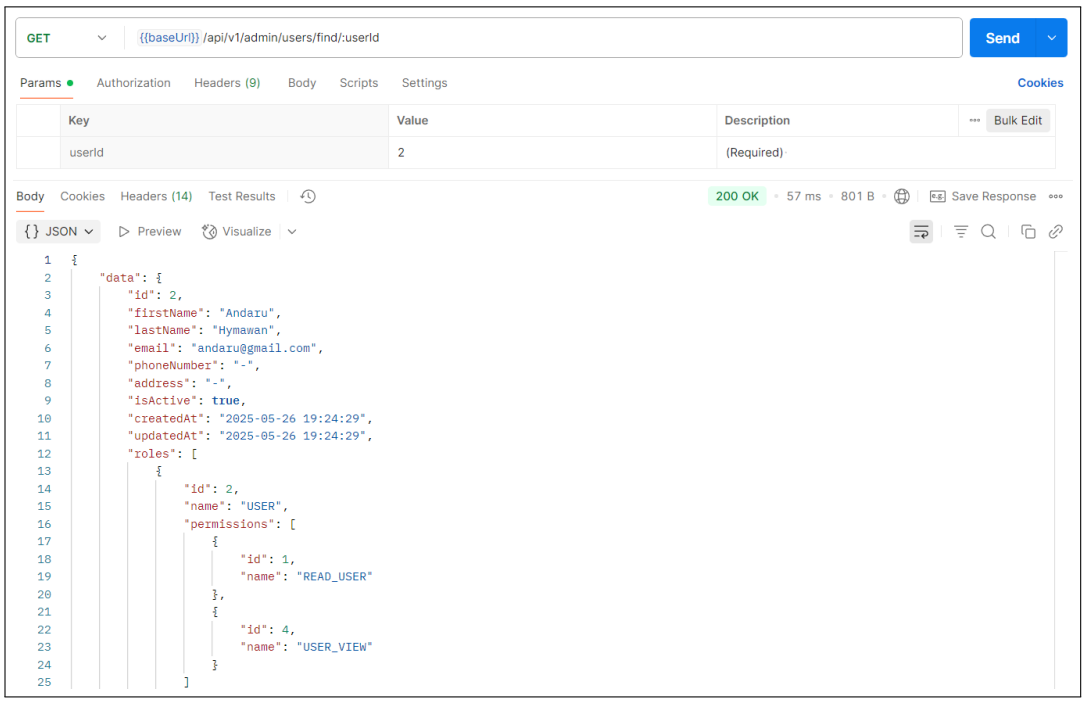
API *Delete Event* berfungsi untuk menghapus data acara spesifik dari sistem berdasarkan `id` acara. Administrator dapat menghapus acara apa pun yang ada

dalam sistem, terlepas dari pembuatnya. Jika penghapusan berhasil, sistem akan mengonfirmasi bahwa data acara telah dihapus. Visualisasi proses ini dapat dilihat pada Gambar 3.31.



Gambar 3.31. Tampilan API Delete Event di Postman

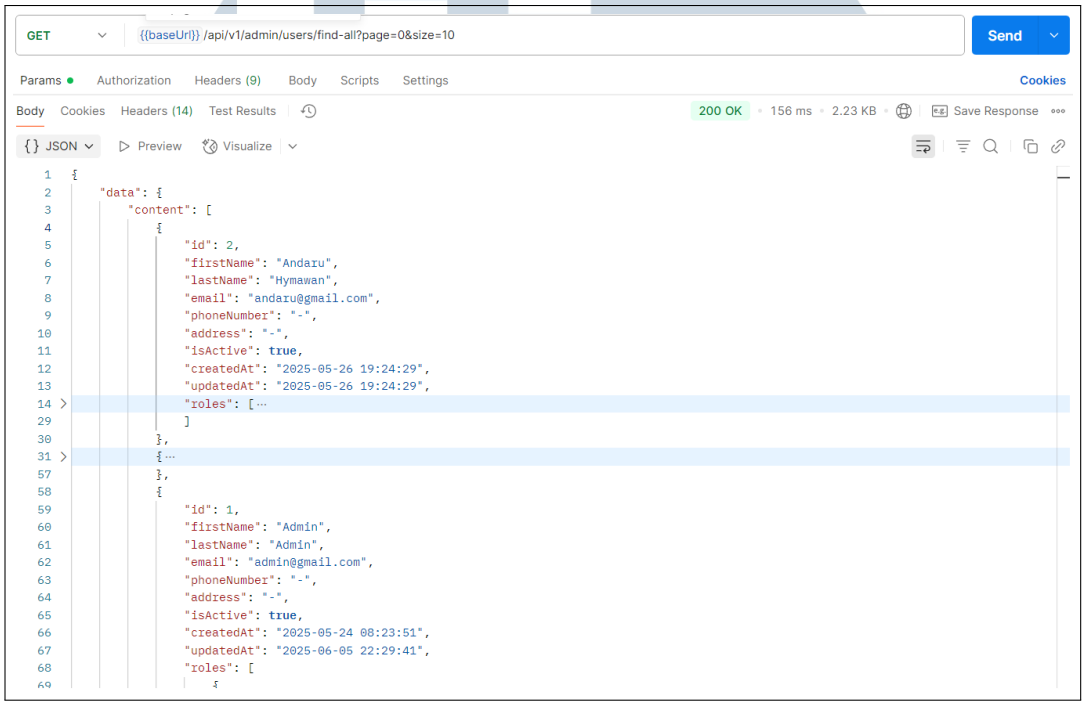
API *Find User* berfungsi untuk mengambil informasi detail mengenai satu pengguna spesifik berdasarkan `id` uniknya. Dengan menyediakan `id` pengguna, administrator dapat memperoleh semua data terkait pengguna tersebut dari sistem. Visualisasi proses ini dapat dilihat pada Gambar 3.32.



Gambar 3.32. Tampilan API Find User di Postman

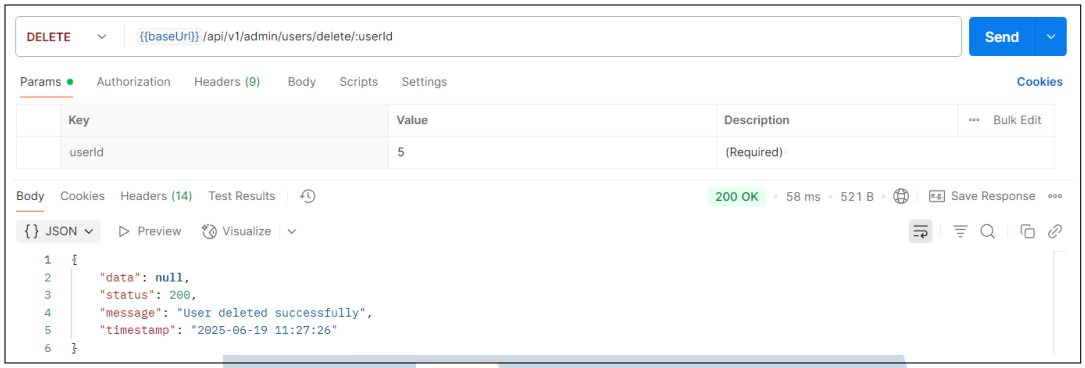
API *Find All Users* memungkinkan administrator untuk mencari dan mengambil

daftar semua pengguna yang terdaftar dalam sistem, dengan dukungan untuk paginasi. Administrator dapat menentukan parameter halaman dan ukuran data per halaman untuk menelusuri daftar pengguna. Hasilnya adalah daftar pengguna lengkap beserta informasi paginasi. Visualisasi proses ini dapat dilihat pada Gambar 3.33.



Gambar 3.33. Tampilan API Find All Users di Postman

API *Delete User* berfungsi untuk menghapus data pengguna spesifik dari sistem berdasarkan `id` pengguna. Proses ini memastikan data pengguna yang tidak lagi diperlukan dapat dikeluarkan dari daftar secara permanen oleh administrator. Jika penghapusan berhasil, sistem akan mengonfirmasi bahwa data pengguna telah dihapus. Visualisasi proses ini dapat dilihat pada Gambar 3.34.



Gambar 3.34. Tampilan API Delete User di Postman

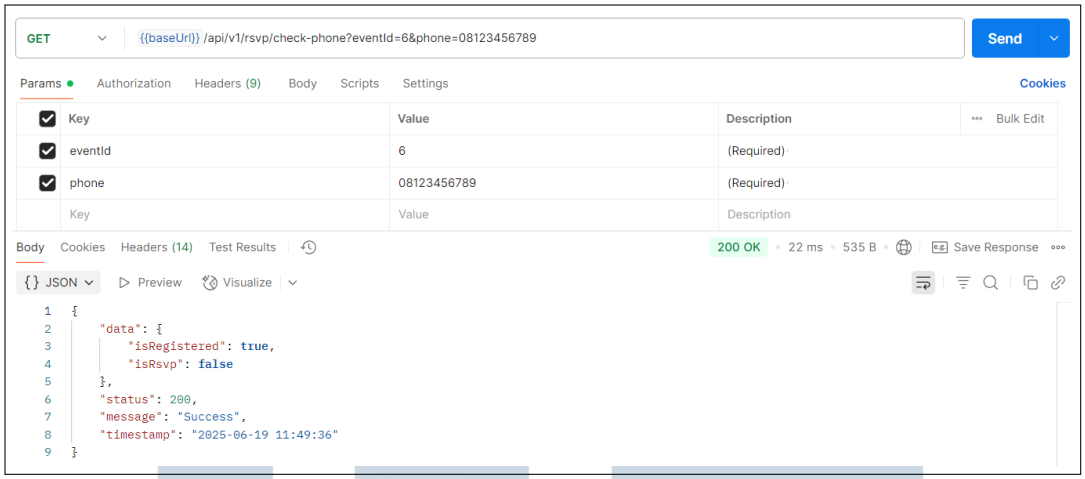
F API RSVP

Bagian ini menjelaskan implementasi API yang terkait dengan mekanisme konfirmasi kehadiran tamu (RSVP) untuk acara. API Konfirmasi Kehadiran memungkinkan tamu untuk mengelola status partisipasi mereka, mulai dari pengecekan nomor telepon hingga konfirmasi kehadiran dan mendapatkan akses masuk melalui kode QR. Rincian mengenai *endpoint* untuk setiap API RSVP dapat dilihat pada Tabel 3.15.

Tabel 3.15. Rincian Endpoint RSVP

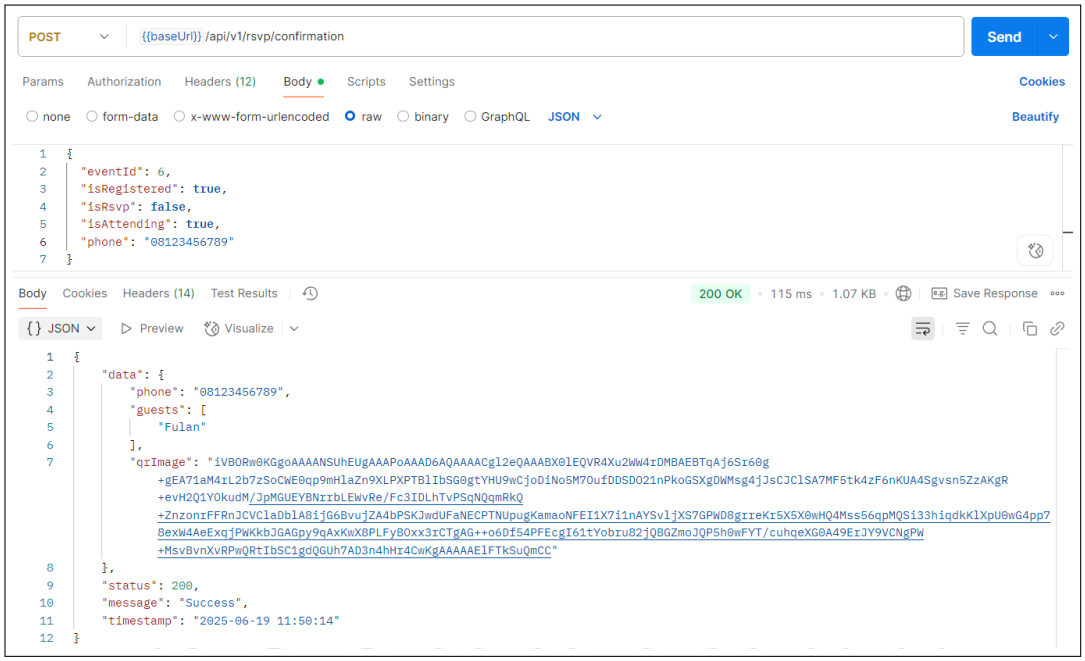
Nama Fitur	Endpoint & Metode HTTP	Deskripsi Singkat
Check Phone	GET /api/v1/rsvp/check-phone	Mencari nomor telepon tamu pada suatu acara
Confirm Guest Attendance	POST /api/v1/rsvp/confirm	Mengonfirmasi kehadiran tamu pada suatu acara

API *Check Phone* berfungsi untuk memverifikasi apakah nomor telepon tertentu sudah terdaftar dalam sistem untuk sebuah acara. Pengguna akan mengirimkan `eventId` dan `phone` yang ingin dicek. Jika `phone` ditemukan, sistem akan memberikan status apakah tamu sudah terdaftar dan sudah melakukan RSVP. Visualisasi proses ini dapat dilihat pada Gambar 3.35.



Gambar 3.35. Tampilan API Check Phone di Postman

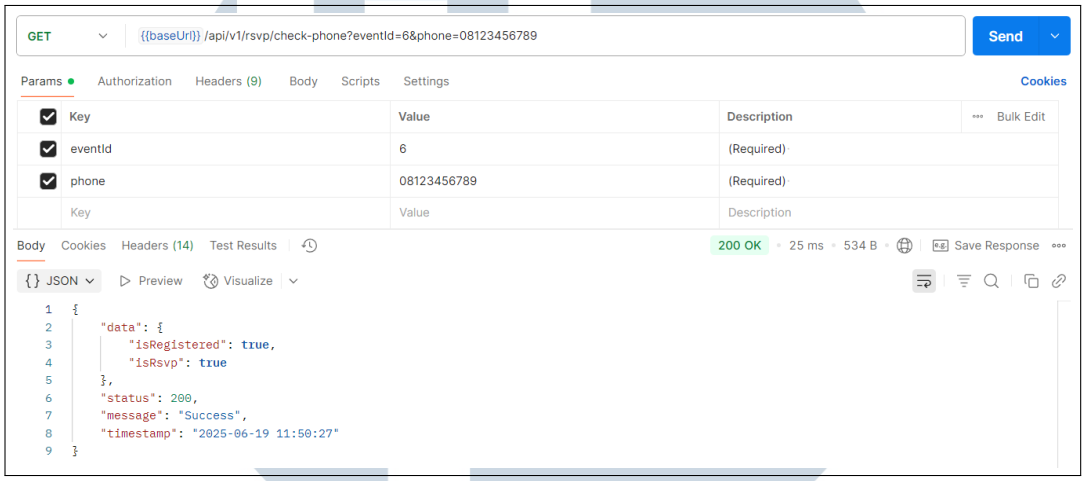
API *Confirm Guest Attendance* memungkinkan tamu untuk mengonfirmasi atau mengubah status kehadiran mereka pada suatu acara. Tamu akan mengirimkan `eventId`, `isAttendning`, dan `phone`. Berdasarkan input dan status sebelumnya, sistem akan memperbarui data tamu, memvalidasi kapasitas acara, dan dapat menghasilkan kode QR sebagai tiket masuk. Visualisasi proses ini dapat dilihat pada Gambar 3.36.



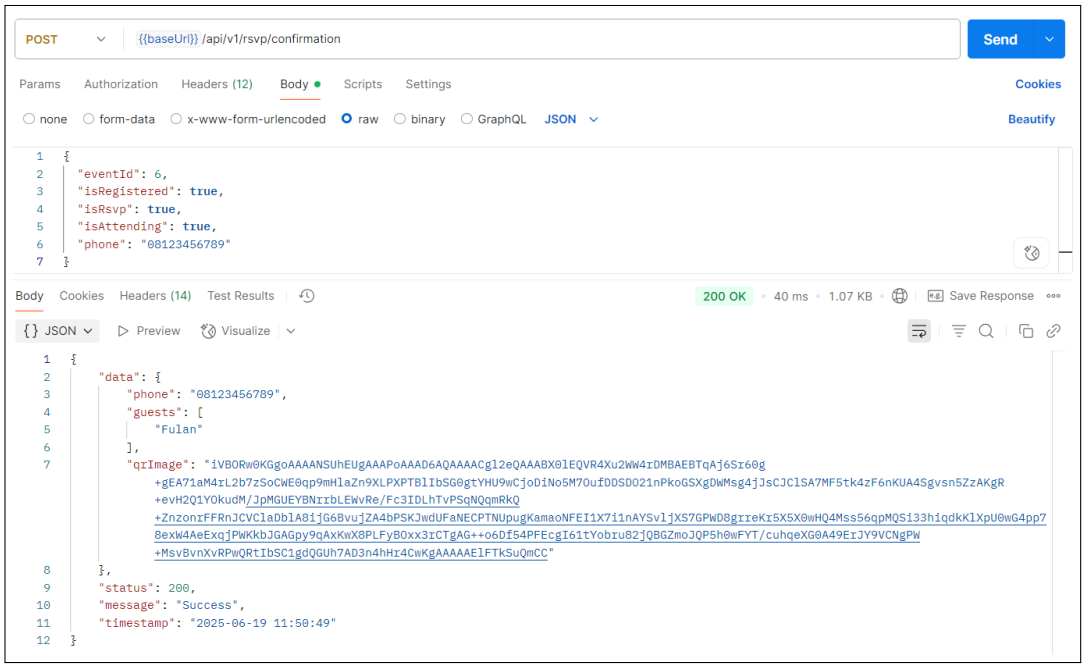
Gambar 3.36. Tampilan API Confirm Guest Attendance di Postman

Jika menggunakan tamu yang sama pada API `Confirm Guest Attendance`, maka

informasi yang dimiliki tamu tersebut sudah diperbarui dan dapat dilihat pada visualisasi proses pada Gambar 3.37. Ketika dicek kembali data tamu tersebut, maka data yang diberikan akan sama persis ketika tamu pertama kali mengisi RSVP. Kesamaan ini dapat dilihat pada Gambar 3.38, di mana data berupa `qr_image` di basis data tidak berubah sama sekali.



Gambar 3.37. Tampilan API Check Phone Jika Tamu Sudah Terdaftar dan Sudah Mengisi RSVP di Postman



Gambar 3.38. Tampilan API Confirm Guest Attendance Dengan Data Tamu yang Sama di Postman

3.3.4 Pengujian API

Bagian ini menguraikan proses pengujian fungsionalitas API yang telah dibangun, yang melibatkan dua fase pengujian utama. Pengujian API ini bertujuan untuk memvalidasi bahwa setiap *endpoint* berfungsi sesuai spesifikasi dan terintegrasi dengan baik.

A Pengujian API oleh Tim Frontend

Pengujian pada fase ini dilakukan oleh tim *frontend* untuk memastikan seluruh API *backend* berfungsi dengan benar dari perspektif klien. Tim *frontend* menguji semua API yang terkait dengan modul Autentikasi, Manajemen Pengguna, Manajemen Acara, dan Manajemen Tamu. Rincian pengujian dan hasilnya dapat dilihat pada Tabel 3.16.

Tabel 3.16. Rincian Pengujian API oleh Tim Frontend

Pengujian	**Hasil yang Diharapkan**	**Hasil Pengujian**	**Kesimpulan**
API Register	Akun pengguna baru berhasil dibuat dengan data valid; pendaftaran gagal jika email duplikat.	Sesuai dengan yang diharapkan.	Sesuai
API Login	Pengguna berhasil login dengan kredensial valid; gagal dengan kredensial tidak valid.	Sesuai dengan yang diharapkan.	Sesuai
API Refresh Token	Token akses baru berhasil didapatkan menggunakan token refresh yang valid; proses gagal jika token kedaluwarsa.	Sesuai dengan yang diharapkan.	Sesuai
API Logout	Sesi pengguna berakhir dan token refresh berhasil diinvalidasi.	Sesuai dengan yang diharapkan.	Sesuai

Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
API Create Event	Acara baru berhasil dibuat dengan detail yang valid.	Sesuai dengan yang diharapkan.	Sesuai
API Find Event	Detail acara spesifik berhasil diambil berdasarkan ID; tidak ditemukan jika ID invalid.	Sesuai dengan yang diharapkan.	Sesuai
API Find All Events	Daftar semua acara berhasil diambil, mendukung paginasi.	Sesuai dengan yang diharapkan.	Sesuai
API Update Event	Detail acara yang sudah ada berhasil diperbarui.	Sesuai dengan yang diharapkan.	Sesuai
API Create Guest	Data tamu tunggal berhasil ditambahkan ke acara dan validasi terkait.	Sesuai dengan yang diharapkan.	Sesuai
API Create Multiple Guest	Sesuai dengan yang diharapkan.	Berhasil	Sesuai
API Find Guest	Detail tamu spesifik berhasil diambil berdasarkan ID yang valid.	Sesuai dengan yang diharapkan.	Sesuai
API Find All Guest (by Event)	Daftar semua tamu untuk acara tertentu berhasil diambil.	Sesuai dengan yang diharapkan.	Sesuai
API Delete Guest	Data tamu spesifik berhasil dihapus.	Sesuai dengan yang diharapkan.	Sesuai

Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
API Manajemen Global	Fungsionalitas global (misal: pengambilan semua data pengguna) dengan hak akses admin berfungsi sesuai.	Sesuai dengan yang diharapkan.	Sesuai

B Pengujian API Konfirmasi Kehadiran (RSVP) oleh Supervisor dan Tim Frontend

Fase pengujian ini secara khusus berfokus pada mekanisme konfirmasi kehadiran tamu (RSVP), yang dilakukan bersama oleh supervisor Jesivinica Christy Santoso dan tim Frontend. Pengujian ini mencakup validasi alur pengecekan status telepon dan proses konfirmasi kehadiran. Rincian pengujian dapat dilihat pada Tabel 3.17.

Tabel 3.17. Rincian Pengujian API Konfirmasi Kehadiran (RSVP) oleh Supervisor dan Tim Frontend

Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Kesimpulan
API Check Phone	Status terdaftar dan status RSVP tamu dapat diverifikasi dengan akurat.	Sesuai dengan yang diharapkan.	Sesuai
API Confirm Guest Attendance	Konfirmasi kehadiran tamu berhasil diproses dan hanya dapat diisi satu kali per tamu/sesi yang sama untuk tujuan integritas data.	Data tamu masih bisa dimasukkan ke basis data berkali-kali walaupun datanya sama.	Kurang Sesuai

3.4 Kendala dan Solusi yang Ditemukan

3.4.1 Kendala

Berdasarkan kerja magang yang dilakukan, terdapat beberapa kendala yang ditemukan, di antaranya:

1. Ketika mencoba mengirim token langsung dari backend ke client melalui Cookie mengalami kendala dikarenakan browser tidak bisa menerima hal tersebut yang padahal jika backend tes langsung di Postman semua aman saja tanpa kendala.
2. Alur sistem RSVP mengalami perombakan besar dan menjadi jauh lebih kompleks dibandingkan desain awal. Hal ini menyulitkan proses penyesuaian, terutama ketika harus menambahkan logika bisnis baru.

3.4.2 Solusi

Berdasarkan kendala yang dialami, berikut merupakan solusinya:

1. Proses pengiriman token diubah dari sebelumnya dikirim langsung oleh *backend* ke klien melalui *cookie*, menjadi melalui *frontend*. Token kini dikirim dari *backend* ke *frontend*, lalu disimpan secara eksplisit oleh *frontend* ke dalam *cookie* atau *localStorage*. Dengan pendekatan ini, *frontend* juga bertanggung jawab mengirimkan token dalam setiap permintaan.
2. Untuk mengakomodasi logika bisnis RSVP yang semakin kompleks, dilakukan restrukturisasi kode dengan menambahkan parameter-parameter baru yang merepresentasikan berbagai kondisi RSVP. Selain itu, penanganan logika dipecah menjadi fungsi-fungsi yang lebih modular dan dikelompokkan berdasarkan kasus, guna mengurangi kompleksitas *if-else* yang berlebihan dan mempermudah pemeliharaan kode.