SISTEM MANAJEMEN KEAMANAN *API KEY* UNTUK *REST API* MENGGUNAKAN *MIDDLEWARE* VALIDASI TOKEN



Disusun Oleh:

N a m a : Riyaldo Dwi Nopianto Suryo Saputro
NIM : 22523139

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
2026

HALAMAN PENGESAHAN DOSEN PEMBIMBING

SISTEM MANAJEMEN KEAMANAN *API KEY* UNTUK *REST API* MENGGUNAKAN *MIDDLEWARE* VALIDASI TOKEN

TUGAS AKHIR JALUR MAGANG

Disusun Oleh:

N a m a : Riyaldo Dwi Nopianto Suryo Saputro
NIM : 22523139

Yogyakarta, 30 Juli 2026

Pembimbing,



(Andhik Budi Cahyono, S.T, M.T)

HALAMAN PENGESAHAN DOSEN PENGUJI

SISTEM MANAJEMEN KEAMANAN *API KEY* UNTUK *REST API* MENGGUNAKAN *MIDDLEWARE* VALIDASI TOKEN

TUGAS AKHIR JALUR MAGANG

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 30 Agustus 2026

Tim Penguji

Andhik Budi Cahyono, S.T., M.T.

Anggota 1

Hari Setiaji, S.Kom., M.Eng.

Anggota 2

Dr. Hendrik, S.T., M.Eng.

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Ir. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., IPM., ASEAN Eng.)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama	:	Riyaldo Dwi Nopianto Suryo Saputro
NIM	:	22523139

Tugas akhir dengan judul:

SISTEM MANAJEMEN KEAMANAN *API KEY* UNTUK *REST API* MENGGUNAKAN *MIDDLEWARE* VALIDASI TOKEN

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 13 Agustus 2026



(Riyaldo Dwi Nopianto Suryo Saputro)

HALAMAN PERSEMBAHAN

Dengan rasa syukur kepada Allah SWT atas segala rahmat dan karunia-Nya, karya ini saya persembahkan kepada:

1. Kedua orang tua tercinta, Bapak dan Ibu, yang selalu memberikan doa, kasih sayang, dukungan, dan pengorbanan yang tidak pernah berhenti dalam setiap langkah kehidupan saya.
2. Kakak yang selalu memberikan semangat, motivasi, dan dukungan selama proses perkuliahan hingga penyusunan laporan ini.
3. Dosen pembimbing dan seluruh dosen Program Studi Informatika yang telah memberikan ilmu, bimbingan, dan arahan selama masa perkuliahan.
4. Sahabat dan teman-teman seperjuangan yang selalu memberikan bantuan, dukungan, serta menemani dalam setiap proses yang telah dilalui.
5. Diri saya sendiri, terima kasih karena telah memilih untuk terus berjalan ketika menyerah terasa lebih mudah.

Semoga karya sederhana ini dapat memberikan manfaat dan menjadi langkah awal untuk terus belajar, berkembang, dan memberikan kontribusi yang lebih baik di masa mendatang.

HALAMAN MOTO

Hanya butuh satu saja satu alasan
Untuk sudah menyerah atau terus berjalan
Syarat pertama adalah percaya,
Syarat kedua, lihat syarat pertama.
— FSTVLST—

KATA PENGANTAR

Assalamu'alaikum Wr. Wb.

Puji syukur penulis panjatkan ke hadirat Allah SWT, karena atas limpahan rahmat, karunia, dan hidayah-Nya, laporan akhir magang yang berjudul "Sistem Manajemen Keamanan *API Key* untuk *REST API* Menggunakan *Middleware* Validasi Token" ini dapat diselesaikan dengan baik. Laporan akhir magang ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana pada Program Studi Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia.

Penulis menyadari bahwa penyusunan laporan akhir magang ini tidak dapat terlaksana dengan baik tanpa adanya dukungan, arahan, serta bantuan dari berbagai pihak. Oleh karena itu, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Allah SWT atas segala rahmat, nikmat, dan hidayah-Nya sehingga penulis diberikan kesehatan, kekuatan, dan kemudahan dalam menyelesaikan kegiatan magang serta penyusunan laporan akhir ini.
2. Kedua orang tua penulis, Bapak Marimin dan Ibu Neti yang senantiasa memberikan doa, kasih sayang, dukungan, dan motivasi sehingga penulis dapat menyelesaikan perkuliahan dan laporan akhir ini dengan baik.
3. Bapak Fathul Wahid selaku Rektor Universitas Islam Indonesia.
4. Bapak Raden Teduh Dirgahayu, S.T., M.Sc., selaku Ketua Jurusan Informatika Universitas Islam Indonesia.
5. Bapak Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D., selaku Ketua Program Studi Informatika Program Sarjana.
6. Bapak Kholid Haryono, S.T., M.Kom., selaku Dosen Pembimbing Akademik yang telah memberikan arahan dan informasi selama proses perkuliahan dan penjaluran magang.
7. Bapak Andhik Budi Cahyono, S.T., M.T., selaku Dosen Pembimbing Laporan Akhir yang telah memberikan bimbingan, arahan, masukan, serta motivasi kepada penulis selama proses penyusunan laporan akhir ini.
8. PT Gamatechno Indonesia beserta seluruh pimpinan dan staff yang telah memberikan kesempatan kepada penulis untuk melaksanakan kegiatan magang, memperoleh pengalaman kerja, serta membantu dalam penyelesaian laporan akhir ini.

9. Saudara kandung penulis, Shinta Apriliana yang telah memberikan dukungan dan motivasi kepada penulis agar menjalankan perkuliahan dengan baik serta menjadi alasan penulis untuk selalu berjalan maju.
10. Teman dekat penulis yaitu Joaquin Brilian, Alfath Rizanatul Ikhsan, Rifa Fahriza, M. Haekal Riza Sampoerna, Rafi Fauzan, Prabawikan, Gilda Re Susanto dan Lutfi Surya Pradana yang telah memberikan dukungan, bantuan, dan kebersamaan selama masa perkuliahan.

Penulis menyadari bahwa laporan akhir magang ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan kritik dan saran yang membangun dari berbagai pihak demi penyempurnaan laporan ini di masa mendatang. penulis berharap semoga laporan akhir magang ini dapat memberikan manfaat serta menambah wawasan dan pengetahuan bagi pembaca, serta dapat menjadi salah satu referensi dalam pengembangan sistem informasi, khususnya pada bidang keamanan layanan *REST API* dan manajemen *API Key*.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, 13 Agustus 2026



(Riyaldo Dwi Nopianto Suryo Saputro)

SARI

Kegiatan magang dilaksanakan di PT Gamatechno Indonesia pada divisi pengembangan perangkat lunak selama enam bulan. Permasalahan yang diidentifikasi selama kegiatan magang adalah belum tersedianya mekanisme autentikasi yang memadai pada layanan *REST API* perusahaan serta tidak adanya sistem pengelolaan token akses yang terpusat, sehingga *endpoint* API dapat diakses oleh pihak mana pun tanpa verifikasi identitas.

Untuk mengatasi permasalahan tersebut, dikembangkan sistem manajemen keamanan *API Key* untuk *REST API* menggunakan *middleware* validasi token berbasis Laravel *Framework*. Sistem yang dibangun mencakup empat komponen utama. Pertama, layanan *REST API* yang menyediakan akses terpadu terhadap data dari tiga subsistem internal perusahaan, yaitu PM, Protrack, dan GES, melalui pendekatan multiple *database* connection dengan total empat belas *endpoint*. Kedua, *middleware CheckApi*Token yang menjalankan empat tahapan validasi secara berurutan, meliputi pemeriksaan keberadaan token, validitas token terhadap basis data SQLite, kesesuaian alamat IP, dan masa berlaku token. Ketiga, *dashboard* administrasi berbasis Filament *Admin panel* yang menyediakan fasilitas pengelolaan *API Key* secara terpusat dengan pembatasan hak akses berbasis peran pengguna. Keempat, mekanisme verifikasi One Time *Password* (OTP) berbasis *session* Laravel untuk mengamankan proses *login* dan *reset password* pada panel administrasi, dengan masa berlaku kode selama lima menit dan mekanisme cooldown 60 detik untuk mencegah penyalahgunaan pengiriman ulang kode.

Proses pengembangan dilakukan menggunakan metodologi Agile, dan pengujian fungsional dilakukan menggunakan Postman terhadap dua puluh skenario pengujian. Hasil pengujian menunjukkan bahwa seluruh skenario menghasilkan keluaran yang sesuai dengan yang diharapkan, membuktikan bahwa sistem yang dikembangkan mampu menerapkan mekanisme keamanan berlapis yang terstruktur dan mendukung pengelolaan akses layanan API perusahaan secara terpusat.

Kata kunci: Laravel, *REST API*, *API Key*, *Middleware*, OTP, SQLite, Filament, Agile, *Backend*.

GLOSARIUM

Agile	Metode pengembangan perangkat lunak yang bersifat iteratif dan fleksibel, sehingga memungkinkan penyesuaian terhadap perubahan kebutuhan selama proses pengembangan berlangsung.
API	Antarmuka yang memungkinkan dua sistem perangkat lunak saling berkomunikasi dan bertukar data melalui sekumpulan aturan dan mekanisme tertentu.
API Key	Serangkaian karakter unik yang digunakan sebagai identitas atau kunci akses untuk mengidentifikasi aplikasi atau pengguna yang mengakses suatu layanan API serta membatasi hak akses terhadap layanan tersebut.
Bearer Token	Token autentikasi yang dikirimkan melalui *header* HTTP Authorization dengan format *Bearer* <token> dan digunakan sebagai bukti identitas untuk mengakses layanan yang dilindungi.
Cooldown	Selang waktu tunggu yang diterapkan sebelum suatu aksi dapat diulang, digunakan untuk membatasi pengiriman ulang kode OTP guna mencegah penyalahgunaan.
Eloquent	Object Relational Mapping (ORM) bawaan Laravel yang memetakan tabel basis data menjadi objek model sehingga interaksi dengan basis data dapat dilakukan tanpa menulis kueri SQL secara langsung.
Endpoint	Alamat spesifik pada layanan *REST API* yang dapat diakses *client* untuk memperoleh atau mengirim data tertentu.
Hash	Hasil pengubahan data menjadi rangkaian karakter tetap melalui fungsi satu arah, digunakan untuk menyimpan kata sandi agar nilai aslinya tidak tersimpan dalam bentuk teks biasa.
JSON	Format pertukaran data berbasis teks yang ringan dan terstruktur, digunakan sebagai format respons pada layanan *REST API*.
Middleware	Lapisan perantara antara *client* dan *controller* yang memproses setiap permintaan sebelum diteruskan,

	digunakan untuk menjalankan validasi keamanan pada layanan *REST API*.
MVC	Pola arsitektur perangkat lunak yang memisahkan pengelolaan data (Model), tampilan (View), dan logika kontrol (*Controller*).
OTP	Kode verifikasi sekali pakai dengan masa berlaku terbatas yang digunakan sebagai lapisan keamanan tambahan pada proses *login* dan *reset password*.
Pagination	Mekanisme pembagian data ke dalam beberapa halaman untuk membatasi jumlah data yang ditampilkan pada setiap permintaan.
Postman	Aplikasi yang digunakan untuk menguji layanan API dengan mensimulasikan berbagai skenario permintaan HTTP.
REST API	Gaya arsitektur layanan API yang memanfaatkan protokol HTTP dan memisahkan tanggung jawab antara penyedia dan konsumen layanan.
Routing	Mekanisme pemetaan alamat permintaan ke fungsi atau *controller* yang menanganinya di dalam *framework*.
Session	Mekanisme penyimpanan data sementara di sisi server selama satu sesi interaksi pengguna, digunakan untuk menyimpan kode OTP selama proses verifikasi.
SQLite	Sistem manajemen basis data relasional ringan berbasis berkas tunggal, digunakan untuk menyimpan data kredensial pengguna dan token *API Key*.

DAFTAR ISI

HALAMAN JUDUL i
HALAMAN PENGESAHAN DOSEN PEMBIMBING ii
HALAMAN PENGESAHAN DOSEN PENGUJI iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR iv
HALAMAN PERSEMBAHAN v
HALAMAN MOTO vi
KATA PENGANTAR vii
SARI ix
GLOSARIUM x
DAFTAR ISI xii
DAFTAR TABEL xiv
DAFTAR GAMBAR xv
BAB I PENDAHULUAN 1
1.1 Latar Belakang 1
1.2 Ruang Lingkup 2
1.3 Batasan Aktivitas 3
1.4 Tujuan 3
1.5 Manfaat 4
1.6 Sistematika Penulisan 4
BAB II LANDASAN TEORI DAN TINJAUAN PUSTAKA 6
2.1 Laravel *Framework* 6
2.2 *REST API* 7
2.3*Middleware CheckApi*Token pada Laravel 9
2.4Manajemen *API Key* 10
2.5Validasi Token 11
2.6 OTP Verification 13
2.7 MySQL 14
2.8 SQLite 14
2.9 Metode Agile 15
2.10 Tinjauan Pustaka 16
BAB III PELAKSANAAN MAGANG 19
3.1 Manajemen Proyek 19
3.1.1 Daftar dan Lini Masa Pengerjaan Proyek 21

3.1.1 Dokumentasi Iterasi Pengembangan 22
3.2 Sistem Manajemen Token API 24
3.2.1 Analisis Permasalahan 24
3.2.2 Perancangan Sistem 25
3.3 Implementasi Sistem 28
3.3.1 Implementasi *REST API* Data Proyek 28
3.3.2 Implementasi *Middleware* Validasi Token 29
3.3.3 Implementasi Sistem Manajemen *API Key* dan *Admin panel* 34
3.3.4 Implementasi Verifikasi OTP dan *Reset password* 37
3.4 Pengujian Sistem 46
BAB IV REFLEKSI PELAKSANAAN MAGANG 49
4.1 Relevansi Akademik 49
4.2 Pembelajaran Magang 51
BAB V PENUTUP 54
5.1 Kesimpulan 54
5.2 Saran 55
DAFTAR PUSTAKA 57
LAMPIRAN 59

DAFTAR TABEL

Tabel 3.1 Daftar dan Lini Masa Pengerjaan Proyek 22
Tabel 3.2 Perancangan *Endpoint REST API* 26
Tabel 3.3 Struktur Tabel api_accesses pada SQLite 27
Tabel 3.4 Hasil Pengujian Sistem 46

DAFTAR GAMBAR

Gambar 2.1 MVC Laravel 6
Gambar 2.2 *REST API* 8
Gambar 2.3 *Middleware* 9
Gambar 2.4 *API Key* 11
Gambar 2.5 Validasi Token 12
Gambar 2.6 OTP Verification 14
Gambar 2.7 Metode Agile 16
Gambar 3.1 Alur Manajemen Proyek 20
Gambar 3.2 Riwayat Commit Git 24
Gambar 3.3 Logbook Aktivitas Harian Magang 24
Gambar 3.4 Respons *Endpoint* GET /api/v1/pm/projects dengan Informasi Pagination 29
Gambar 3.5 Respons *Endpoint* GET /api/v1/pm/projects/{id} dengan Relasi Data Klien 29
Gambar 3.6 Diagram *Middleware* Validasi Token 30
Gambar 3.7 Token API Tidak Valid 32
Gambar 3.8 Token API Tidak Valid 33
Gambar 3.9 IP Tidak Diizinkan 34
Gambar 3.10 Token Kedaluwarsa 34
Gambar 3.11 *Dashboard* Token 35
Gambar 3.12 Pembuatan Token 36
Gambar 3.13 Halaman *Login* 37
Gambar 3.14 *Dashboard User* 37
Gambar 3.15 Kode Program Pengiriman OTP 39
Gambar 3.16 Kode Email OTP 39
Gambar 3.17 Verifikasi Akun dengan OTP 40
Gambar 3.18 Kode Program Mekanisme Cooldown Pengiriman Ulang OTP 41
Gambar 3.19 Kode Program Logika Validasi Kode OTP 42
Gambar 3.20 Notifikasi Kode OTP Salah 44
Gambar 3.21 Tampilan Email *Reset password* 45
Gambar 3.22 Tampilan Pembuatan Kata Sandi Baru 45

BAB I
PENDAHULUAN

1.1 Latar Belakang

PT Gamatechno Indonesia merupakan perusahaan yang bergerak di bidang teknologi informasi dan komunikasi dengan fokus pada pengembangan solusi digital, sistem informasi, serta layanan transformasi digital. Perusahaan ini berfokus pada pengembangan teknologi yang mampu membantu instansi maupun perusahaan dalam meningkatkan efisiensi, integrasi sistem, serta kualitas pelayanan melalui pemanfaatan teknologi informasi. Hal ini menjadikan solusi yang dikembangkan lebih adaptif terhadap kebutuhan pengguna dan mengikuti perkembangan teknologi terkini. Dengan pengalaman dalam menangani berbagai proyek nasional dan sistem berskala besar, PT Gamatechno Indonesia menjadi salah satu perusahaan yang memberikan kesempatan bagi penulis untuk mempelajari proses pengembangan perangkat lunak secara nyata di lingkungan industri.

Dalam menjalankan aktivitas pengembangan sistemnya, PT Gamatechno Indonesia banyak mengerjakan proyek yang melibatkan pertukaran data antar aplikasi melalui pemanggilan *REST API*. Layanan API menjadi komponen penting dalam menghubungkan berbagai subsistem internal perusahaan maupun integrasi dengan sistem pihak ketiga. Namun, pada praktiknya, pengelolaan akses terhadap layanan API tersebut belum sepenuhnya terstruktur. Beberapa permasalahan yang teridentifikasi antara lain belum adanya mekanisme autentikasi yang memadai pada sebagian *endpoint* API, tidak adanya pengelolaan terpusat terhadap kunci akses yang diterbitkan, serta tidak tersedianya kontrol atas masa berlaku dan asal alamat IP pengakses layanan. Kondisi ini berpotensi menimbulkan risiko keamanan, seperti akses tidak sah terhadap data internal perusahaan dan penyalahgunaan kunci akses yang telah didistribusikan.

Berdasarkan permasalahan tersebut, diperlukan mekanisme pengamanan akses API yang mampu memastikan bahwa hanya pihak yang memiliki otorisasi sah yang dapat mengakses layanan, sekaligus memberikan kontrol kepada administrator dalam mengelola siklus hidup setiap kunci akses. Untuk menjawab kebutuhan ini, dirancang sebuah sistem manajemen *API Key* berbasis Laravel *Framework* yang dilengkapi dengan *middleware* validasi token, pembatasan akses berdasarkan alamat IP, pengaturan masa berlaku token, serta panel administrasi berbasis Filament untuk mempermudah pengelolaan kunci akses oleh administrator. Pemilihan Laravel didasarkan pada ketersediaan fitur *middleware*, sistem

routing yang terstruktur, serta dukungan ekosistem yang memadai untuk pengembangan layanan *REST API* yang aman dan mudah dipelihara.

Proyek ini secara langsung menjawab permasalahan pengelolaan akses API yang sebelumnya telah diuraikan. Tantangan teknis yang perlu diselesaikan secara sistematis dalam proyek ini meliputi validasi token, pembatasan akses berdasarkan alamat IP, pengelolaan masa berlaku kunci akses, serta pengamanan *endpoint* dari akses yang tidak sah. Untuk menjawab kebutuhan tersebut, proyek ini berfokus pada implementasi *REST API* menggunakan Laravel *Framework* dengan menerapkan sistem manajemen *API Key*, *middleware* validasi token, pembatasan akses berbasis IP Address, *admin panel* menggunakan Filament, serta fitur keamanan pengguna berupa *reset password* dan verifikasi OTP saat *login*. Pengujian terhadap seluruh *endpoint* juga dilakukan menggunakan Postman untuk memastikan sistem berjalan sesuai kebutuhan.

Melalui proyek ini diharapkan dapat dihasilkan sebuah sistem API yang terstruktur, aman, dan mudah dikembangkan lebih lanjut. Selain memberikan kontribusi langsung bagi kebutuhan internal perusahaan, pengalaman yang diperoleh selama pengerjaan proyek ini juga diharapkan dapat menjadi referensi dalam pengembangan *backend* service berbasis Laravel, khususnya dalam aspek implementasi keamanan dan manajemen akses API pada lingkungan enterprise.

1.2 Ruang Lingkup

Kegiatan magang dilaksanakan selama enam bulan di PT Gamatechno Indonesia pada divisi pengembangan perangkat lunak. Selama pelaksanaan magang, penulis terlibat dalam beberapa proyek pengembangan perangkat lunak, seperti pengembangan *dashboard* berbasis web dan integrasi sistem Enterprise Resource Planning (ERP) berbasis Odoo. Namun, mengingat keterkaitannya dengan tema dan tujuan laporan, pembahasan pada laporan ini difokuskan pada proyek Pengembangan Sistem Manajemen Keamanan *API Key* untuk *REST API* Menggunakan *Middleware* Validasi Token yang dikembangkan menggunakan *framework* Laravel.

Adapun ruang lingkup pengembangan sistem yang dibahas dalam laporan ini meliputi:

1. Pengembangan *REST API* untuk menyediakan data proyek dengan mekanisme pagination.
2. Implementasi sistem manajemen *API Key* menggunakan basis data SQLite.

3. Implementasi *middleware* validasi token untuk melakukan autentikasi dan otorisasi akses API.
4. Implementasi pembatasan akses berdasarkan alamat IP.
5. Implementasi pemeriksaan masa berlaku *API Key*.
6. Pengembangan *admin panel* berbasis Filament untuk pengelolaan *API Key*.
7. Implementasi autentikasi dua faktor melalui verifikasi One Time *Password* (OTP) saat *login* administrator.
8. Implementasi fitur *reset password* melalui email menggunakan mekanisme bawaan Filament.
9. Pengujian fungsional terhadap fitur keamanan dan pengelolaan *API Key* yang dikembangkan.

1.3 Batasan Aktivitas

Pembahasan dalam laporan ini difokuskan pada pengembangan sistem *backend* berbasis Laravel *Framework* yang mencakup implementasi layanan *REST API* untuk penyediaan data proyek, sistem manajemen *API Key*, *middleware* validasi token, pembatasan akses berdasarkan alamat IP, pengaturan masa berlaku token, pengembangan *admin panel* menggunakan Filament, serta fitur keamanan pengguna berupa *reset password* melalui email dan verifikasi OTP saat *login*. Pengujian sistem dilakukan menggunakan Postman sebatas pada fungsionalitas *endpoint* dan mekanisme keamanan yang telah diimplementasikan.

Laporan ini tidak membahas pengembangan antarmuka *frontend* secara mendalam, karena sistem yang dikembangkan berfungsi sebagai layanan *backend* yang dapat dikonsumsi oleh berbagai klien secara independen. Selain itu, proyek *dashboard* berbasis Node.js dan proyek ERP berbasis Odoo yang dikerjakan selama periode magang tidak dibahas secara mendetail, karena kedua proyek tersebut bersifat sebagai pengalaman pendukung dan bukan merupakan fokus utama laporan. Proses *deployment* ke server produksi juga tidak termasuk dalam cakupan pembahasan laporan ini, karena pengembangan dilakukan dalam lingkungan lokal dan jaringan internal perusahaan.

1.4 Tujuan

Tujuan dari pengembangan sistem manajemen *API Key* dan keamanan akses *REST API* pada kegiatan magang ini adalah:

1. Membangun sistem manajemen *API Key* berbasis Laravel *Framework* yang mampu mengontrol token akses, mencakup pembuatan, pengelolaan masa berlaku, dan pencabutan *API Key* secara terpusat melalui *admin panel* berbasis Filament.
2. Mengimplementasikan mekanisme keamanan berlapis pada layanan *REST API* melalui *middleware* validasi token, pembatasan akses berbasis IP Address, dan pemeriksaan masa berlaku token, sehingga hanya permintaan yang memenuhi seluruh persyaratan keamanan yang dapat mengakses *endpoint* API.
3. Menyediakan layanan *REST API* yang terstruktur untuk mengintegrasikan data dari tiga subsistem internal perusahaan, yaitu PM, Protrack, dan GES, sehingga data dari masing-masing subsistem dapat diakses melalui endpoint API yang telah disediakan.
4. Meningkatkan keamanan proses autentikasi pengguna pada *admin panel* melalui penerapan verifikasi OTP saat *login* dan fitur *reset password* berbasis email, sehingga akses ke sistem pengelolaan *API Key* hanya dapat dilakukan oleh pengguna yang berwenang.

1.5 Manfaat

Berikut merupakan manfaat yang dapat diperoleh dari pengembangan sistem manajemen *API Key* dan keamanan akses *REST API* ini:

1. Bagi PT Gamatechno Indonesia, sistem yang dikembangkan memberikan mekanisme pengelolaan dan pengamanan akses layanan *REST API* yang lebih terstruktur. Dengan adanya sistem manajemen *API Key*, administrator dapat mengontrol siapa yang berhak mengakses layanan API, berapa lama akses berlaku, serta dari alamat IP mana akses diizinkan, sehingga risiko akses tidak sah terhadap data internal perusahaan dapat diminimalkan.
2. Bagi penulis, kegiatan pengembangan sistem ini memberikan pengalaman langsung dalam mengimplementasikan keamanan *REST API* secara berlapis menggunakan Laravel *Framework*, Filament, dan Postman pada lingkungan pengembangan perangkat lunak industri yang sesungguhnya.

1.6 Sistematika Penulisan

Penyajian laporan magang ini menggunakan sistematika penulisan sebagai berikut:

BAB I PENDAHULUAN

Bab ini membahas mengenai latar belakang penulisan laporan tugas akhir, ruang lingkup, batasan aktivitas, tujuan, manfaat, dan sistematika penulisan laporan tugas akhir.

BAB II LANDASAN TEORI DAN TINJAUAN PUSTAKA

Bab ini membahas mengenai landasan teori yang berkaitan dengan pengembangan *REST API*, keamanan akses berbasis *API Key*, *middleware*, serta hasil penelitian terdahulu yang relevan dengan topik yang dibahas.

BAB III PELAKSANAAN MAGANG

Bab ini membahas tentang aktivitas yang dilakukan selama pelaksanaan magang di PT Gamatechno Indonesia, penjelasan proyek-proyek yang dikerjakan, serta pembahasan implementasi sistem manajemen *API Key* dan keamanan *REST API* menggunakan Laravel *Framework*.

BAB IV REFLEKSI PELAKSANAAN MAGANG

Bab ini membahas tentang analisis relevansi antara teori akademik dengan pelaksanaan di lapangan, pembelajaran yang diperoleh selama magang, serta kendala, hambatan, dan tantangan yang dihadapi selama kegiatan magang berlangsung.

BAB V PENUTUP

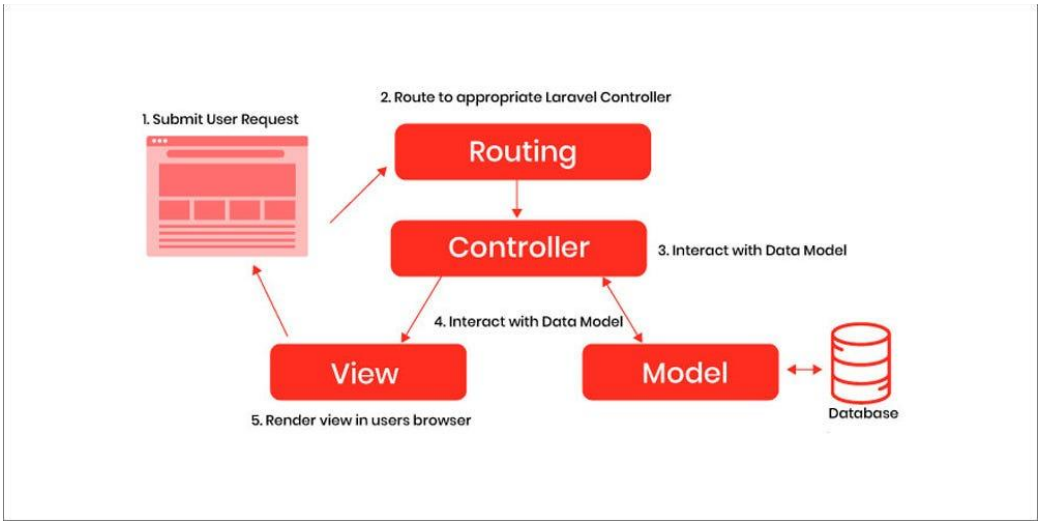
Bab ini membahas kesimpulan dari kegiatan magang yang telah dilaksanakan serta saran yang dapat diberikan untuk pengembangan sistem maupun kegiatan magang di masa mendatang.

BAB II
LANDASAN TEORI DAN TINJAUAN PUSTAKA

2.1 Laravel *Framework*

Laravel merupakan *framework* pengembangan aplikasi web berbasis PHP yang bersifat open-source dan diciptakan oleh Taylor Otwell untuk menyederhanakan proses pengembangan aplikasi web modern melalui penyediaan berbagai fitur yang mendukung pembangunan aplikasi secara lebih terstruktur dan efisien (Herdiyatmoko, 2022).

Framework Laravel menerapkan pola arsitektur Model View *Controller* (MVC) yang memisahkan pengembangan aplikasi ke dalam tiga komponen utama, yaitu Model, View, dan *Controller*. Model bertugas mengelola data dan logika bisnis yang berhubungan dengan basis data, View berfungsi menampilkan antarmuka kepada pengguna, sedangkan *Controller* bertindak sebagai penghubung antara Model dan View dengan memproses permintaan pengguna dan menentukan respons yang akan dikembalikan (Rahmawati & Sumarsono, 2024). Penerapan arsitektur MVC membantu menghasilkan struktur kode yang lebih terorganisir, mudah dipelihara, dan mempermudah proses pengembangan aplikasi dalam jangka panjang. Arsitektur MVC pada Laravel ditunjukkan pada Gambar 2.1, yang memperlihatkan alur interaksi antara Model, View, dan Controller dalam memproses permintaan pengguna.



Gambar 2.1 MVC Laravel

Sumber: ritik065.medium.com

Selain menerapkan pola MVC, Laravel juga menyediakan sistem routing yang fleksibel untuk mengatur hubungan antara *Uniform Resource Locator* (URL) dengan fungsi atau *controller* yang akan dijalankan. Fitur routing memungkinkan pengembang mengelola alur permintaan dan respons aplikasi secara lebih terstruktur (Rahmawati & Sumarsono, 2024).

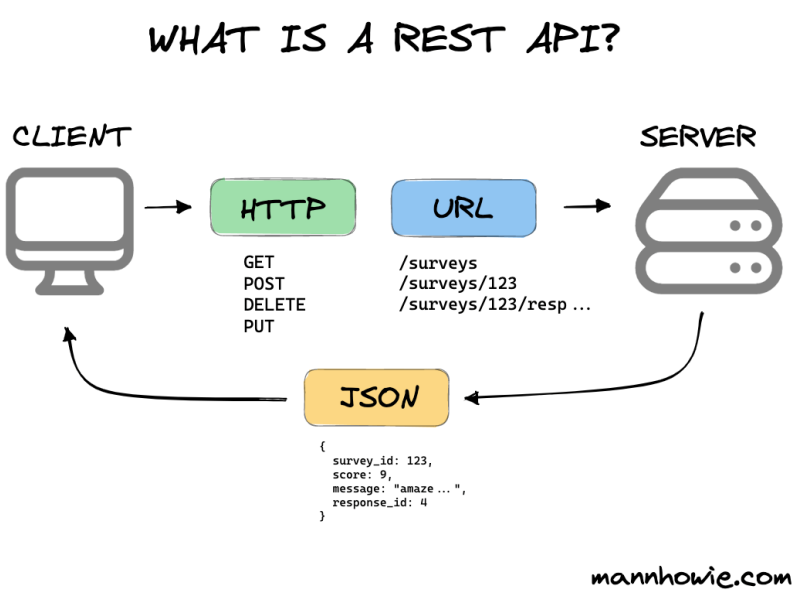
Dalam hal pengelolaan basis data, Laravel menyediakan library *Eloquent Object Relational Mapping* (ORM) yang membantu dan mempermudah proses pengolahan data secara langsung menggunakan Class Model (Arianto & Susetyo, 2022). Penggunaan *Eloquent ORM* membantu pengembang dalam mengelola data secara lebih terstruktur dan terintegrasi dengan komponen aplikasi lainnya.

Pada pengembangan layanan berbasis *REST API*, Laravel juga menyediakan fitur *middleware* yang berfungsi sebagai lapisan perantara dalam menangani permintaan (*request*) dan respons (*response*). *Middleware* memungkinkan sistem melakukan proses pemeriksaan tertentu sebelum permintaan diteruskan ke logika utama aplikasi, seperti proses autentikasi, otorisasi, dan validasi token. Dengan demikian, hanya pengguna atau aplikasi yang memenuhi persyaratan keamanan yang dapat mengakses layanan yang disediakan (Arianto & Susetyo, 2022).

Pengembangan aplikasi modern juga banyak menerapkan konsep layered architecture atau arsitektur berlapis, yaitu pemisahan tanggung jawab sistem ke dalam beberapa lapisan, seperti presentation layer, business logic layer, dan data layer. Pemisahan tersebut bertujuan meningkatkan modularitas, mempermudah proses pemeliharaan, serta membuat sistem lebih adaptif terhadap perubahan kebutuhan di masa mendatang (Rachman et al., 2025). Mengadopsi prinsip serupa, pola MVC pada Laravel juga menerapkan pemisahan tanggung jawab antarkomponen sehingga *framework* ini menjadi andal dalam pengembangan layanan *backend* dan *REST API* yang terstruktur serta mudah dikembangkan.

2.2 *REST API*

Representational State Transfer (REST) merupakan gaya arsitektur yang digunakan dalam penerapan web service dan banyak digunakan karena kesederhanaannya dalam membangun layanan berbasis web (Rizal & Rahmatulloh, 2019). *REST API* memungkinkan pertukaran data antara *client* dan server melalui protokol HTTP serta mendukung integrasi antar sistem yang berbeda. Ilustrasi komunikasi antara client dan server pada arsitektur REST API ditunjukkan pada Gambar 2.2.



Gambar 2.2 *REST API*

Sumber: mannhowie.com

Komunikasi antara *client* dan server pada *REST API* dilakukan menggunakan metode standar HTTP, yaitu GET untuk mengambil data, POST untuk menambahkan data, PUT atau PATCH untuk memperbarui data, dan DELETE untuk menghapus data. Format pertukaran data yang umum digunakan adalah JavaScript Object Notation (JSON) karena format tersebut memudahkan penerima dalam membaca dan memahami data yang dipertukarkan (Rizal & Rahmatulloh, 2019), serta secara sintaksis kompatibel dengan pengembangan aplikasi web modern (Herdiyatmoko, 2022).

Salah satu karakteristik utama *REST API* adalah sifatnya yang stateless, yaitu server tidak menyimpan informasi status sesi pengguna sehingga setiap permintaan dari *client* harus bersifat independen dan membawa seluruh informasi yang diperlukan untuk diproses oleh server. Karakteristik ini memungkinkan layanan REST menjadi lebih sederhana dan mempermudah proses integrasi antar sistem (Rizal & Rahmatulloh, 2019).

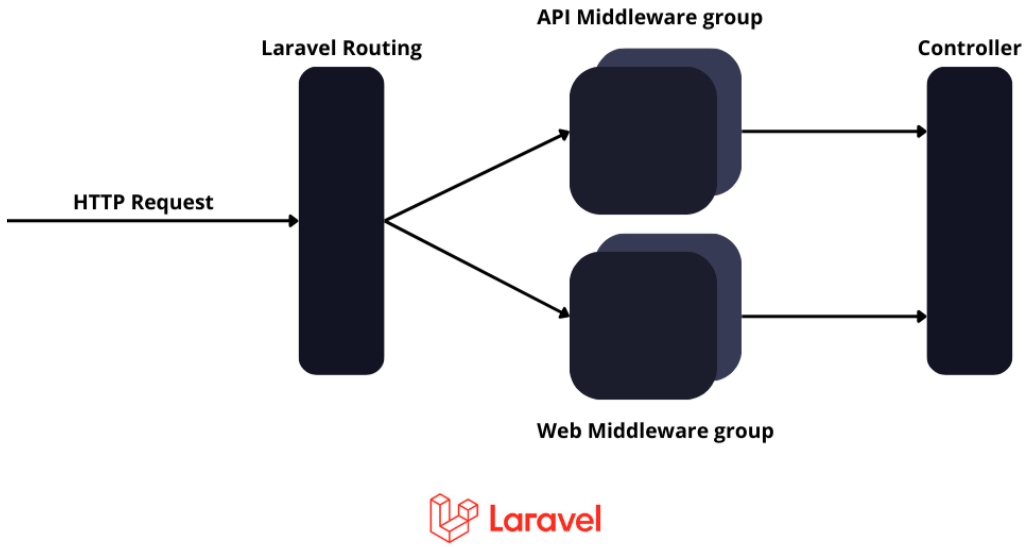
Dalam pengamanan layanan *REST API*, mekanisme autentikasi umumnya dilakukan menggunakan token. Setelah proses autentikasi berhasil dilakukan, sistem akan menghasilkan Bearer Token yang harus disertakan pada parameter Authorization untuk setiap pertukaran informasi antara *client* dan server (Arianto & Susetyo, 2022).

Validasi token merupakan proses pemeriksaan keabsahan token sebelum sistem memberikan akses terhadap suatu layanan. Proses validasi mencakup pemeriksaan kredensial

pengguna, pembatasan ruang lingkup akses (scope), serta otorisasi pengguna. Penerapan protokol seperti OAuth2 memungkinkan sistem menolak permintaan yang tidak sah dengan mengembalikan respons HTTP yang sesuai, seperti 401 Unauthorized apabila kredensial tidak valid atau 400 *Bad Request* apabila parameter dan hak akses yang dikirimkan tidak sesuai (Ramadhan Purba et al., 2025).

2.3 *Middleware CheckApi*Token pada Laravel

Middleware merupakan mekanisme yang digunakan untuk memproses permintaan (*request*) sebelum diteruskan ke *controller* dan memproses *respons* (*response*) sebelum dikembalikan kepada pengguna. Pada *framework* Laravel, *middleware* banyak dimanfaatkan untuk menerapkan berbagai mekanisme keamanan, seperti autentikasi, otorisasi, dan pembatasan akses terhadap layanan tertentu. Posisi middleware dalam alur pemrosesan permintaan pada Laravel ditunjukkan pada Gambar 2.3.



Gambar 2.3 *Middleware*
Sumber: buildwithangga.com

Pada pengembangan *REST API*, *middleware* dapat digunakan untuk memeriksa keberadaan dan keabsahan token autentikasi sebelum permintaan diproses lebih lanjut. Penerapan *middleware* untuk autentikasi memungkinkan sistem melakukan penyaringan akses

berdasarkan token dan hak akses pengguna sehingga hanya permintaan yang memenuhi ketentuan keamanan yang dapat diteruskan ke aplikasi (Arianto & Susetyo, 2022).

Dalam implementasi Laravel, *middleware* dapat dikustomisasi sesuai kebutuhan aplikasi. Salah satu implementasinya adalah *middleware CheckApi*Token, yaitu *middleware* yang dikembangkan untuk melakukan pemeriksaan terhadap token yang dikirimkan pada *header* permintaan. *Middleware* ini bertugas memvalidasi keberadaan token, membandingkan token yang diterima dengan token yang tersimpan pada sistem, serta menolak permintaan yang tidak memenuhi ketentuan autentikasi.

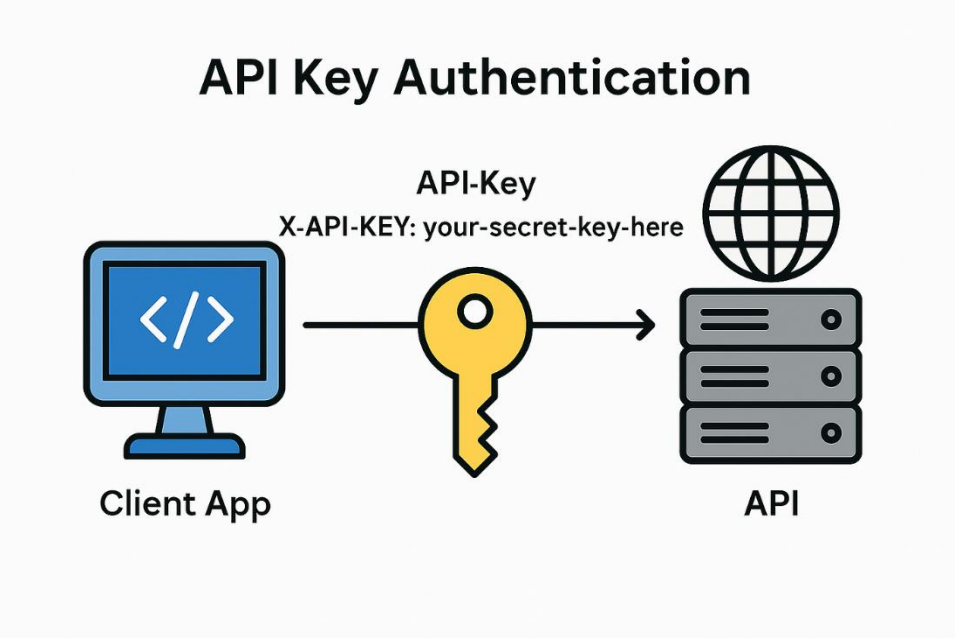
Apabila token tidak ditemukan atau tidak sesuai dengan data yang tersimpan, sistem akan menolak permintaan dan mengembalikan pesan kesalahan berupa Unauthenticated, sehingga layanan API hanya dapat diakses oleh pengguna atau aplikasi yang memiliki token yang valid (Ndaru et al., 2025). Pada implementasi lain, kegagalan autentikasi juga dapat ditunjukkan melalui respons HTTP 401 Unauthorized sebagai indikasi bahwa kredensial yang dikirimkan tidak sah (Ramadhan Purba et al., 2025).

Dengan demikian, penggunaan *middleware CheckApi*Token dapat meningkatkan keamanan layanan *REST API* karena proses validasi dilakukan sebelum permintaan mencapai logika utama aplikasi, sehingga risiko akses oleh pihak yang tidak berwenang dapat diminimalkan.

2.4 Manajemen *API Key*

API Key merupakan kunci akses yang digunakan untuk mengidentifikasi dan mengontrol akses suatu aplikasi terhadap layanan Application Programming Interface (API). Dalam implementasi layanan berbasis *REST API*, mekanisme pengamanan sering kali menggunakan token autentikasi yang dikirimkan pada *header* permintaan sebagai bukti bahwa permintaan berasal dari pengguna atau aplikasi yang memiliki hak akses.

Pada implementasi RESTful web service menggunakan Laravel, proses autentikasi menghasilkan Bearer Token yang harus disertakan pada parameter Authorization setiap kali *client* melakukan pertukaran informasi dengan server (Arianto & Susetyo, 2022). Konsep tersebut sejalan dengan penggunaan *API Key* dalam penelitian ini, di mana setiap permintaan yang masuk diwajibkan menyertakan kunci akses yang valid sebelum layanan API dapat diproses lebih lanjut. Ilustrasi mekanisme penggunaan API Key sebagai kunci akses layanan ditunjukkan pada Gambar 2.4.



Gambar 2.4 *API Key*

Sumber: medium.com

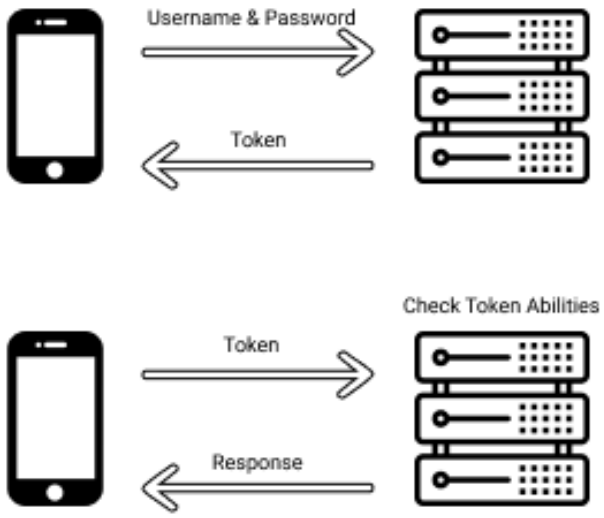
Penggunaan token atau kunci akses yang dipadukan dengan *middleware* memungkinkan sistem melakukan proses penyaringan terhadap setiap permintaan yang masuk. Permintaan yang tidak menyertakan token yang valid atau tidak memenuhi aturan autentikasi akan ditolak sehingga hanya pengguna atau aplikasi yang memiliki hak akses yang dapat menggunakan layanan API (Ndaru et al., 2025).

Pengamanan layanan API tidak hanya berfokus pada proses autentikasi dan otorisasi, tetapi juga perlu memperhatikan aspek keamanan lainnya, seperti keamanan transport layer, validasi input, serta pemantauan aktivitas sistem secara berkelanjutan (Ramadhan Purba et al., 2025). Penerapan berbagai mekanisme keamanan tersebut bertujuan untuk menjaga kerahasiaan data, mencegah akses oleh pihak yang tidak berwenang, dan meningkatkan keandalan layanan API yang dikembangkan.

2.5 Validasi Token

Token autentikasi merupakan identitas digital yang diterbitkan oleh sistem kepada pengguna atau aplikasi setelah proses autentikasi berhasil dilakukan. Pada implementasi RESTful web service menggunakan Laravel, sistem akan menghasilkan Bearer Token yang harus disertakan pada parameter Authorization setiap kali *client* melakukan pertukaran

informasi dengan server (Arianto & Susetyo, 2022). Ilustrasi proses penerbitan dan penggunaan token autentikasi ditunjukkan pada Gambar 2.5.



Gambar 2.5 Validasi Token
Sumber: blog.akhdani.co.id

Validasi token merupakan proses pemeriksaan keabsahan token sebelum sistem memberikan akses terhadap suatu layanan. Proses validasi dilakukan untuk memastikan bahwa token yang dikirimkan oleh pengguna atau aplikasi memenuhi ketentuan autentikasi dan otorisasi yang telah ditetapkan. Penerapan protokol seperti OAuth2 memungkinkan sistem menolak permintaan yang tidak sah dengan mengembalikan respons HTTP yang sesuai, seperti 401 Unauthorized apabila kredensial tidak valid atau 400 Bad *Request* apabila parameter dan hak akses yang dikirimkan tidak sesuai (Ramadhan Purba et al., 2025).

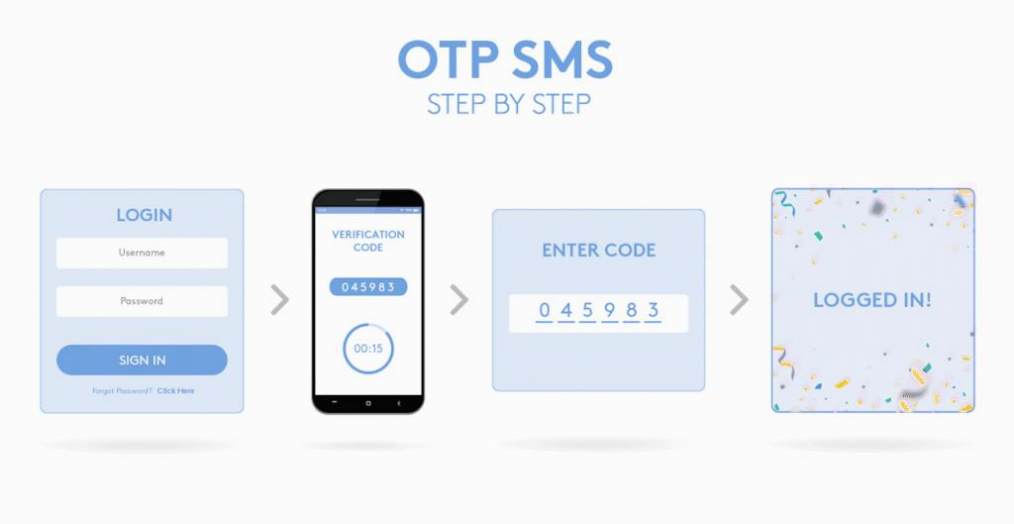
Pada proyek magang ini, validasi token diterapkan sebagai lapisan keamanan utama pada *endpoint REST API*. Setiap permintaan yang masuk akan diperiksa keberadaan token dan kesesuaiannya dengan token yang tersimpan pada sistem sebelum permintaan diproses lebih lanjut. Pendekatan ini memastikan bahwa hanya permintaan dengan token yang sah yang dapat mengakses layanan API sehingga keamanan sistem dapat ditingkatkan.

2.6 OTP Verification

One-Time *Password* (OTP) adalah mekanisme autentikasi yang menghasilkan kode verifikasi unik yang hanya dapat digunakan satu kali dalam jangka waktu tertentu. OTP umumnya berupa kombinasi angka atau huruf acak yang dikirimkan kepada pengguna melalui media tertentu seperti SMS, email, aplikasi autentikator, atau aplikasi pesan instan. Karena bersifat sementara dan dinamis, OTP sulit untuk dieksploitasi oleh pihak yang tidak berwenang meskipun mereka berhasil mendapatkan kata sandi utama pengguna (Nurandani et al., 2025).

OTP berfungsi sebagai lapisan verifikasi tambahan yang memastikan bahwa pengguna yang melakukan akses benar-benar pemilik akun yang sah. Setiap kode OTP umumnya disertai dengan masa berlaku yang singkat, sehingga apabila kode tidak digunakan dalam jangka waktu tersebut, kode akan secara otomatis kedaluwarsa dan tidak dapat dipakai kembali. Karakteristik ini meningkatkan keamanan sistem karena kode yang berhasil disadap menjadi tidak bernilai setelah masa berlakunya habis atau setelah digunakan satu kali (Nurandani et al., 2025).

OTP merupakan salah satu komponen utama dalam penerapan *Two Factor Authentication* (2FA), yaitu metode autentikasi yang menggabungkan dua faktor berbeda untuk memverifikasi identitas pengguna. Faktor pertama umumnya berupa sesuatu yang diketahui pengguna seperti kata sandi, sedangkan faktor kedua berupa sesuatu yang dimiliki pengguna secara fisik seperti perangkat penerima OTP. Penerapan 2FA dengan OTP terbukti secara signifikan mengurangi risiko akses tidak sah, bahkan dalam kondisi kata sandi pengguna telah bocor (Nurandani et al., 2025). Ilustrasi proses verifikasi menggunakan kode OTP ditunjukkan pada Gambar 2.6.



Gambar 2.6 OTP Verification
Sumber: decisiontele.com

Pada proyek magang ini, OTP diterapkan sebagai mekanisme verifikasi tambahan pada proses *login* pengguna serta pada proses *reset password*. Dengan adanya OTP, sistem memiliki lapisan keamanan kedua yang memperkuat perlindungan terhadap akun pengguna dari potensi akses yang tidak sah.

2.7 MySQL

MySQL merupakan sistem manajemen basis data relasional yang banyak digunakan dalam pengembangan aplikasi berbasis web karena mampu menyimpan dan mengelola data secara terstruktur. Pada pengembangan aplikasi *backend*, MySQL sering digunakan sebagai media penyimpanan data utama yang terintegrasi dengan *framework* web modern, termasuk Laravel (Ramadhan Purba et al., 2025).

Pada *framework* Laravel, proses interaksi dengan basis data dapat dilakukan menggunakan library Eloquent Object Relational Mapping (ORM). Library ini membantu dan mempermudah proses pengolahan data melalui penggunaan Class Model sehingga pengembang dapat mengelola data secara lebih terstruktur dan terintegrasi dengan komponen aplikasi lainnya (Arianto & Susetyo, 2022).

Kombinasi Laravel dan MySQL banyak digunakan dalam pengembangan layanan *REST API* karena mampu mendukung proses pengelolaan data secara efektif dan mempermudah integrasi antara aplikasi dan basis data (Ramadhan Purba et al., 2025).

Pada proyek magang ini, MySQL digunakan sebagai basis data utama yang menyimpan data operasional aplikasi, seperti data proyek, data pengguna, data token, serta berbagai data pendukung lainnya yang dibutuhkan oleh sistem. Seluruh data tersebut dikelola melalui mekanisme akses basis data pada Laravel sehingga proses penyimpanan, pengambilan, dan pembaruan data dapat dilakukan secara terstruktur dan konsisten.

2.8 SQLite

SQLite adalah *Relational Database Management System* (RDBMS) yang bersifat serverless, self-contained, dan open source. Sistem ini banyak digunakan sebagai penyimpanan data relasional lokal karena sifatnya yang ringan dan sederhana, serta mendukung prinsip ACID, sehingga andal dalam menjaga integritas data (Rachman et al., 2025;Hermanto, Utama,

Muflihah, Januantoro, & Kusnanto, 2024). Selain itu, SQLite merupakan *database* gratis yang dapat digunakan dengan mudah untuk membuat dan mengelola basis data serta tidak menerapkan arsitektur *client*-server seperti sistem manajemen basis data pada umumnya (Febryana et al., 2024).

SQLite memiliki karakteristik yang sederhana namun cukup fungsional. Sistem ini mendukung sebagian besar fitur SQL standar dan dapat berjalan pada berbagai sistem operasi tanpa memerlukan instalasi yang rumit. Karena sifatnya yang berbasis file, SQLite tidak memerlukan koneksi jaringan untuk diakses, sehingga proses pembacaan dan penulisan data menjadi sangat cepat untuk skala data yang relatif kecil hingga menengah.

Kelebihan utama SQLite adalah portabilitas, kesederhanaan, dan tingkat ketergantungan yang rendah terhadap infrastruktur tambahan. Sistem ini menjadi pilihan utama untuk aplikasi yang memerlukan basis data lokal karena sifatnya yang ringan dan kemudahan dalam integrasi ke dalam aplikasi (Rachman et al., 2025). SQLite banyak digunakan pada aplikasi mobile, aplikasi desktop, sistem embedded, maupun sebagai basis data pendukung pada aplikasi web. Perbedaan utama SQLite dengan MySQL terletak pada arsitekturnya, MySQL bersifat berbasis server dan dirancang untuk skala enterprise dengan banyak pengguna konkuren, sementara SQLite bersifat serverless dan dirancang untuk kebutuhan yang lebih sederhana dengan beban kerja yang relatif ringan.

Pada proyek magang ini, SQLite dipilih sebagai basis data terpisah yang dikhususkan untuk menyimpan data *API Key* beserta konfigurasi pendukungnya. Pemilihan SQLite didasarkan pada karakteristik data *API Key* yang relatif ringan dan tidak memerlukan infrastruktur basis data yang kompleks. Pemisahan ini juga memungkinkan subsistem keamanan API dapat berjalan secara mandiri terlepas dari kondisi basis data utama.

2.9 Metode Agile

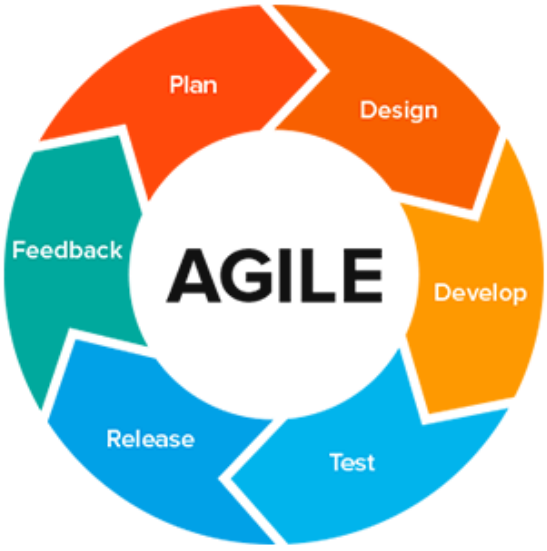
Agile merupakan metode pengembangan perangkat lunak yang menekankan fleksibilitas, kolaborasi tim, serta kemampuan untuk merespons perubahan kebutuhan selama proses pengembangan berlangsung. Pendekatan ini dilakukan secara iteratif dan inkremental sehingga pengembangan sistem dapat dilakukan secara bertahap melalui siklus-siklus pendek (Zahra et al., 2024).

Pada metode Agile, proses pengembangan dilakukan melalui interaksi yang berulang antara tim pengembang dan pengguna untuk memperoleh umpan balik secara berkelanjutan. Penambahan fitur yang kemudian diuji dan dievaluasi sebelum memasuki tahap berikutnya.

Pendekatan ini memungkinkan tim untuk melakukan penyesuaian terhadap kebutuhan sistem yang dapat berubah selama proses pengembangan (Zahra et al., 2024).

Salah satu implementasi Agile yang banyak digunakan adalah Scrum. Scrum membagi proses pengembangan ke dalam beberapa tahapan yang dilakukan secara berulang, mulai dari perencanaan kebutuhan, pengembangan fitur, pengujian, hingga evaluasi hasil pengembangan. Melalui proses tersebut, tim dapat melakukan perbaikan secara berkelanjutan dan menghasilkan perangkat lunak yang lebih sesuai dengan kebutuhan pengguna (Zahra et al., 2024).

Penerapan metode Agile memberikan berbagai manfaat, seperti mempermudah tim dalam menghadapi perubahan kebutuhan, meningkatkan komunikasi dan kolaborasi antaranggota tim, serta membantu mengurangi risiko kesalahan karena proses pengembangan dan evaluasi dilakukan secara bertahap pada setiap iterasi. Ilustrasi siklus iteratif pada metode Agile ditunjukkan pada Gambar 2.7.



Gambar 2.7 Metode Agile

Sumber: aguayo.co

2.10 Tinjauan Pustaka

Beberapa penelitian terdahulu yang relevan dengan pengembangan sistem keamanan *REST API* dan pengelolaan autentikasi diuraikan sebagai berikut.

Penelitian pertama dilakukan oleh (Herdiyatmoko, 2022) dengan judul *Desain Sistem Backend Berbasis REST API Menggunakan Framework Laravel 7*. Penelitian ini

mengembangkan sistem *backend* berbasis *REST API* menggunakan Laravel 7 dengan pola arsitektur MVC, memanfaatkan fitur *Database* Migration sebagai version control basis data dan Laravel Artisan sebagai generator otomatis. Hasil pengujian menunjukkan bahwa seluruh metode HTTP berhasil dieksekusi dengan kode status HTTP 200 pada seluruh skenario pengujian. Penelitian ini belum menerapkan mekanisme autentikasi maupun pembatasan akses pada *endpoint REST API*.

Penelitian kedua dilakukan oleh (Ndaru et al., 2025) dengan judul *Implementasi REST API Pengelolaan Data Penduduk Multi-Desa Berbasis Laravel dengan Autentikasi Sanctum*. Penelitian ini menerapkan teknik pagination untuk pengelolaan data serta menggunakan Laravel Sanctum sebagai mekanisme autentikasi pada *REST API*. Hasil penelitian menunjukkan bahwa sistem dapat memblokir permintaan yang tidak menyertakan token autentikasi. Penelitian ini belum membahas pengelolaan token secara terpusat maupun penerapan pembatasan akses berbasis alamat IP.

Penelitian ketiga dilakukan oleh (Nurandani et al., 2025) dengan judul *Penggunaan OTP (One Time Password) Sebagai Lapisan Keamanan Tambahan dalam Aplikasi AY Pulsa*. Penelitian ini mengimplementasikan autentikasi dua faktor menggunakan OTP berbasis PIN dinamis yang didistribusikan melalui WhatsApp API. Hasil penelitian menunjukkan bahwa penggunaan OTP dapat meningkatkan keamanan proses autentikasi pengguna. Namun, implementasi OTP pada penelitian ini difokuskan pada autentikasi pengguna aplikasi dan belum diterapkan pada pengamanan layanan *REST API*.

Selain merujuk pada penelitian terdahulu, tinjauan pustaka pada laporan ini juga mengacu pada standar *OWASP API Security Top 10*, yaitu daftar risiko keamanan API yang paling kritis menurut *Open Worldwide Application Security Project* (OWASP). Beberapa risiko yang relevan dengan sistem yang dikembangkan pada laporan ini antara lain *Broken Authentication* dan *Broken Object Level Authorization*, yang masing-masing dijawab pada implementasi sistem melalui mekanisme validasi token pada *middleware CheckApiToken* serta pembatasan akses berdasarkan alamat IP dan masa berlaku token. Standar praktik penggunaan token pembawa (*Bearer Token*) yang diterapkan pada sistem ini juga sejalan dengan spesifikasi otorisasi berbasis token dalam ekosistem HTTP sebagaimana didefinisikan dalam RFC 6750.

Berdasarkan ketiga penelitian tersebut, terdapat beberapa aspek yang belum ditangani secara bersamaan. Penelitian pertama belum menerapkan autentikasi sama sekali, penelitian kedua menambahkan autentikasi namun belum menyediakan pengelolaan token terpusat dan pembatasan berbasis IP, sedangkan penelitian ketiga menerapkan OTP tetapi hanya pada

autentikasi pengguna aplikasi dan tidak terintegrasi dengan pengamanan layanan API. Sistem yang dikembangkan dalam laporan magang ini mengisi celah tersebut dengan mengintegrasikan *middleware* validasi token kustom, *dashboard* administrasi berbasis Filament untuk pengelolaan *API Key* secara terpusat, pembatasan akses berbasis alamat IP, serta verifikasi OTP berbasis email untuk mengamankan akses panel administrasi.

BAB III
PELAKSANAAN MAGANG

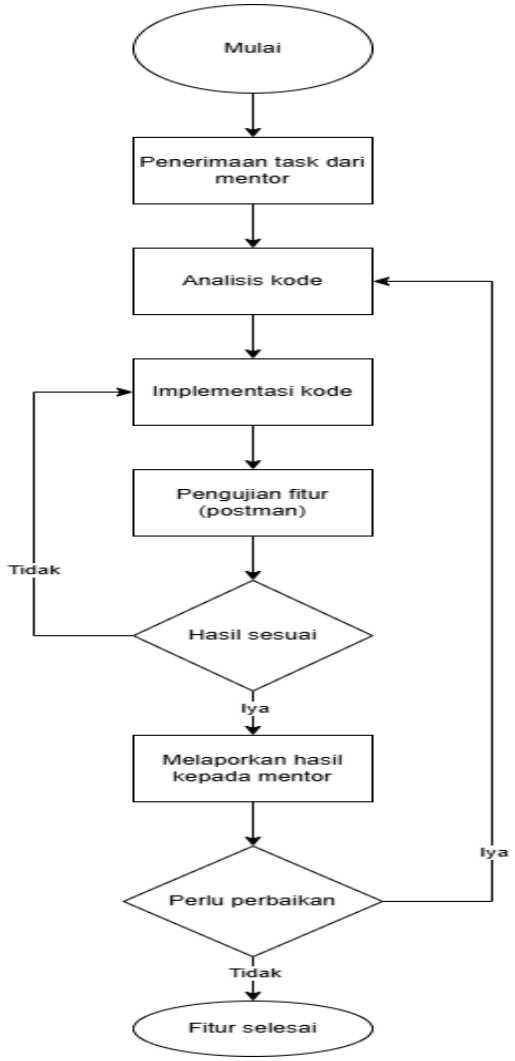
3.1 Manajemen Proyek

Pelaksanaan magang di PT Gamatechno Indonesia difokuskan pada pengembangan sistem *backend* berbasis Laravel *Framework* yang mencakup layanan *REST API*, sistem manajemen *API Key*, serta mekanisme keamanan akses. Proses pengembangan dilakukan menggunakan metodologi Agile dengan pendekatan pengembangan iteratif yang memungkinkan penyesuaian terhadap perubahan kebutuhan sistem secara dinamis tanpa mengganggu keseluruhan alur pengembangan yang sedang berjalan.

Metodologi Agile digunakan sebagai pendekatan pengembangan yang menekankan proses pengerjaan secara bertahap, pengujian, evaluasi, serta perbaikan berdasarkan kebutuhan yang ditemukan selama proses pengembangan. Pendekatan ini sesuai dengan kondisi pengerjaan selama pelaksanaan magang karena tugas dan kebutuhan sistem dapat mengalami perubahan berdasarkan hasil evaluasi mentor. Setiap pekerjaan tidak selalu diselesaikan dalam satu tahap, tetapi dapat mengalami perbaikan atau penyesuaian sebelum dinyatakan selesai. Dengan demikian, penerapan Agile pada proyek ini berfokus pada prinsip pengembangan secara iteratif dan adaptif, tanpa menerapkan *framework* Scrum atau Kanban secara formal.

Secara teknis, penugasan pada pelaksanaan magang disampaikan oleh mentor melalui pesan WhatsApp tanpa menggunakan tools manajemen proyek formal seperti Trello atau Jira. Setiap tugas yang diterima dikerjakan sesuai dengan kebutuhan yang diberikan. Setelah implementasi selesai, hasil pekerjaan diuji dan dilaporkan kepada mentor untuk memperoleh evaluasi dan umpan balik. Kode yang telah selesai kemudian dikirimkan (push) ke repositori Git sebagai bagian dari proses integrasi dan dokumentasi perubahan kode. Apabila terdapat perbaikan atau perubahan kebutuhan berdasarkan hasil evaluasi, pekerjaan kembali dilakukan melalui tahap analisis, implementasi, dan pengujian. Siklus tersebut berlangsung berulang selama proses pengembangan proyek.

Pada penerapannya di lingkungan PT Gamatechno Indonesia, alur manajemen proyek dilaksanakan melalui tahapan yang terstruktur, sebagaimana ditunjukkan pada Gambar 3.1. Gambar tersebut memperlihatkan tahapan mulai dari penerimaan tugas, analisis sistem, proses pengembangan, pengujian, hingga penyelesaian pekerjaan.



Gambar 3.1 Alur Manajemen Proyek

Tahap pertama adalah penerimaan dan analisis tugas. Setiap pengembang memperoleh tugas dalam bentuk task yang diberikan oleh mentor atau supervisor melalui sistem manajemen tugas internal perusahaan. Setiap task memuat spesifikasi fungsional yang harus diimplementasikan, baik berupa pembuatan fitur baru, optimasi kode yang sudah ada, maupun perbaikan kesalahan sistem. Pada tahap ini, penulis melakukan pembacaan dan pemahaman terhadap spesifikasi task secara menyeluruh sebelum memulai proses implementasi, untuk memastikan bahwa fitur yang akan dikembangkan sesuai dengan ekspektasi dan kebutuhan sistem yang telah ditetapkan.

Tahap kedua adalah analisis kode dan struktur sistem yang sedang berjalan. Sebelum menulis kode baru, penulis terlebih dahulu mempelajari struktur kode, skema basis data, serta alur logika bisnis pada sistem yang sudah ada. Tahap ini penting untuk memastikan bahwa fitur baru yang akan ditambahkan dapat terintegrasi dengan modul-modul yang telah berjalan tanpa

menimbulkan konflik atau kerusakan pada bagian sistem lainnya. Proses analisis ini juga mencakup pemahaman terhadap konvensi penulisan kode yang digunakan dalam proyek, seperti penamaan variabel, struktur *controller*, dan pola desain yang diterapkan, sehingga kode yang dihasilkan tetap konsisten dengan keseluruhan basis kode proyek.

Tahap ketiga adalah proses pengembangan atau implementasi. Pada tahap ini, penulis melakukan penulisan kode program pada lingkungan pengembangan lokal dengan memanfaatkan Laravel *Framework* sebagai fondasi utama pengembangan *backend*. Seluruh fitur dikembangkan mengikuti pola arsitektur Model View *Controller* (MVC) yang diterapkan oleh Laravel. Untuk fitur-fitur yang berkaitan dengan antarmuka administrasi, pengembangan dilakukan menggunakan Filament *Admin panel* yang memungkinkan pembuatan tampilan pengelolaan data secara cepat dan terstandar. Setiap perubahan kode yang dilakukan juga disertai dengan pembaruan pada konfigurasi dan dokumentasi teknis yang diperlukan agar memudahkan proses pemeliharaan di kemudian hari.

Tahap keempat adalah pengujian fitur. Setelah implementasi selesai dilakukan, setiap fitur diuji secara langsung pada lingkungan lokal menggunakan Postman. Pengujian mencakup verifikasi terhadap format respons JSON yang dihasilkan, ketepatan status kode HTTP yang dikembalikan pada setiap skenario permintaan, serta validasi terhadap mekanisme keamanan yang telah diimplementasikan seperti penolakan akses pada token tidak valid, token kedaluwarsa, maupun permintaan dari alamat IP yang tidak terdaftar. Apabila hasil pengujian tidak sesuai dengan spesifikasi yang ditetapkan, penulis melakukan perbaikan dan mengulangi proses pengujian hingga seluruh skenario menghasilkan respons yang diharapkan.

Setelah seluruh tahapan tersebut diselesaikan, hasil pengembangan dilaporkan kepada mentor untuk dievaluasi. Apabila terdapat perbaikan atau penambahan kebutuhan berdasarkan hasil evaluasi, task yang bersangkutan dikembalikan ke tahap analisis dan proses iterasi diulangi kembali. Siklus ini berlangsung secara berulang hingga seluruh fitur dinyatakan selesai dan memenuhi standar yang ditetapkan oleh perusahaan.

3.1.1 Daftar dan Lini Masa Pengerjaan Proyek

Selama enam bulan pelaksanaan magang di divisi pengembangan perangkat lunak PT Gamatechno Indonesia, penulis mengerjakan tiga proyek dengan tingkat kompleksitas yang meningkat secara bertahap. Proyek pertama berfungsi sebagai pengenalan terhadap alur kerja perusahaan, sedangkan proyek ketiga merupakan proyek utama yang menjadi fokus

pembahasan laporan ini. Daftar proyek beserta lini masa pengerjaannya disajikan pada Tabel 3.1.

Tabel 3.1 Daftar dan Lini Masa Pengerjaan Proyek

No	Nama Proyek	Teknologi	Periode	Hasil
1	Pengembangan *Dashboard* BantuUMKM.com	React, Node.js	Bulan ke 1 sampai bulan ke 2	*Dashboard* antarmuka pengelolaan data UMKM sebagai proyek pengenalan alur kerja pengembangan
2	Kustomisasi Modul ERP Ganesha Operation	Odoo	Bulan ke 2 sampai bulan ke 5	Penyesuaian modul manajemen anggaran sesuai kebutuhan operasional internal
3	Sistem Manajemen *API Key* dan Layanan *REST API*	Laravel, Filament, SQLite, MySQL	Bulan ke 5 sampai bulan ke 6	Sistem manajemen token akses terpusat, 14 *endpoint REST API*, *middleware* keamanan, dan fitur OTP

Berdasarkan Tabel 3.1, pembahasan dalam laporan ini difokuskan pada proyek ketiga, yaitu Sistem Manajemen *API Key* dan Layanan *REST API*. Proyek tersebut dipilih karena merupakan proyek utama yang dikerjakan pada tahap akhir pelaksanaan magang serta mencakup pengembangan sistem secara menyeluruh, mulai dari implementasi layanan *REST API* hingga penerapan mekanisme keamanan berbasis *API Key* dan OTP.

3.1.1 Dokumentasi Iterasi Pengembangan

Penerapan prinsip Agile pada pengerjaan proyek Sistem Manajemen API Key tidak menggunakan artefak Scrum formal seperti product backlog, user story, sprint planning, atau sprint retrospective. Selama pelaksanaan magang, penugasan disampaikan oleh mentor lapangan melalui WhatsApp. Setiap tugas kemudian dianalisis, diimplementasikan, diuji, dan dilaporkan kepada mentor untuk memperoleh evaluasi. Apabila terdapat perubahan kebutuhan atau hasil pekerjaan belum sesuai, perbaikan dilakukan dengan mengulangi tahap analisis, implementasi, dan pengujian. Dengan demikian, penerapan Agile pada proyek ini lebih tepat dijelaskan sebagai siklus pengembangan iteratif berbasis task dan umpan balik, bukan sebagai penerapan Scrum secara formal.

Periode pengembangan pertama (29 Januari–2 Februari 2026) berfokus pada orientasi terhadap proyek Laravel. Aktivitas yang dilakukan meliputi pembelajaran mengenai Laravel,

middleware, dan API Key, penyambungan proyek dengan basis data perusahaan melalui OpenVPN, penambahan SQLite untuk penyimpanan API Key, serta pemahaman terhadap proyek REST API yang menggunakan *middleware* validasi token.

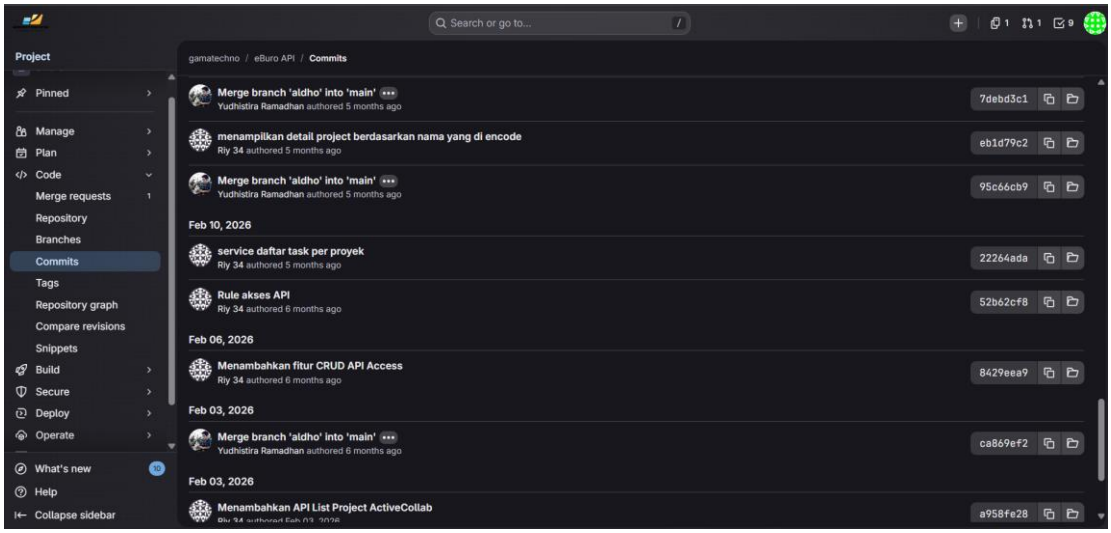
Periode pengembangan kedua (3–13 Februari 2026) berfokus pada pengembangan layanan REST API dan manajemen API Key. Aktivitas yang dilakukan meliputi pemahaman struktur proyek dan alur REST API, pembuatan service untuk detail proyek dan informasi client, pengembangan fitur CRUD API Key, penerapan pembatasan berdasarkan token, alamat IP, dan tanggal kedaluwarsa, implementasi validasi token, pengujian menggunakan Postman, perbaikan pagination, serta pengembangan service untuk data task berdasarkan proyek.

Periode pengembangan ketiga (16–25 Februari 2026) berfokus pada pengembangan dan penyempurnaan fitur verifikasi email dan OTP. Aktivitas yang dilakukan meliputi implementasi pengiriman dan validasi kode OTP, pengujian pengiriman email menggunakan MailHog, perbaikan halaman verifikasi, penyempurnaan tampilan, penambahan mekanisme cooldown pengiriman ulang OTP selama 60 detik, perbaikan tombol kirim ulang, hingga finalisasi dan push setelah memperoleh persetujuan mentor.

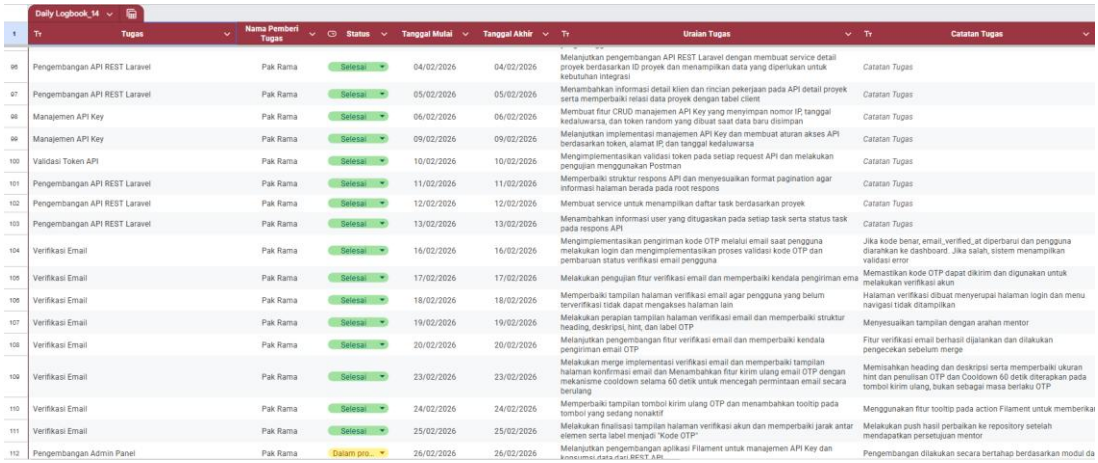
Periode pengembangan keempat pada 26 Februari 2026 dan berfokus pada pengembangan admin panel menggunakan Filament. Aktivitas tersebut meliputi pengembangan aplikasi Filament untuk mendukung pengelolaan sistem yang telah dibangun.

Setelah pekerjaan atau kelompok tugas selesai, hasil pengembangan dilaporkan kepada mentor untuk memperoleh evaluasi dan umpan balik. Apabila terdapat perbaikan atau perubahan kebutuhan, pekerjaan kembali dilakukan melalui tahap analisis, implementasi, dan pengujian. Pola tersebut menunjukkan adanya siklus pengembangan iteratif selama pengerjaan proyek, karena hasil pekerjaan dapat mengalami perubahan berdasarkan evaluasi sebelum dinyatakan selesai.

Sebagai bukti pendukung, dokumentasi proses pengembangan ditelusuri melalui dua sumber. Pertama, riwayat commit dan push pada repositori Git proyek Sistem Manajemen API Key, sebagaimana ditunjukkan pada Gambar 3.2. Riwayat tersebut digunakan untuk menunjukkan jejak perubahan kode dan perkembangan pekerjaan selama proses pengembangan. Kedua, catatan logbook aktivitas harian magang yang mencantumkan tanggal dan uraian tugas yang dikerjakan, sebagaimana ditunjukkan pada Gambar 3.3. Menampilkan aktivitas yang dilakukan pada pengembangan proyek.



Gambar 3.2 Riwayat Commit Git



Gambar 3.3 Logbook Aktivitas Harian Magang

3.2 Sistem Manajemen Token API

Sistem Manajemen Token API merupakan proyek utama yang dikembangkan penulis selama magang dan menjadi fokus pembahasan laporan ini. Sistem dibangun untuk menjawab kebutuhan perusahaan akan mekanisme akses data antar subsistem internal yang aman dan terkelola secara terpusat. Pembahasan pada subbab berikut mencakup analisis kebutuhan, perancangan sistem, implementasi, hingga pengujian sistem.

3.2.1 Analisis Permasalahan

Berdasarkan hasil analisis dan diskusi bersama mentor, ditemukan bahwa layanan *REST API* yang tersedia pada sistem internal perusahaan belum memiliki mekanisme autentikasi yang memadai sehingga dapat diakses oleh pihak mana pun tanpa verifikasi identitas. Kondisi ini

menimbulkan risiko signifikan karena data proyek internal yang bersifat sensitif berpotensi diakses pihak yang tidak berwenang, yang dapat berdampak pada kebocoran informasi perusahaan.

Kebutuhan Fungsional

Kebutuhan fungsional menjelaskan fungsi yang harus disediakan oleh sistem untuk pengguna. Berdasarkan hasil analisis permasalahan, kebutuhan fungsional sistem yang dikembangkan meliputi:

1. Sistem harus menyediakan layanan *REST API* untuk mengakses data proyek dan data pendukung dari berbagai subsistem perusahaan
2. Sistem harus menyediakan mekanisme autentikasi berbasis *API Key* untuk mengamankan akses layanan API
3. Sistem harus mampu memvalidasi masa berlaku *API Key* sebelum memberikan akses ke layanan
4. Sistem harus mampu membatasi akses berdasarkan alamat IP yang telah ditentukan
5. Sistem harus menyediakan *dashboard* administrasi untuk mengelola *API Key*
6. Sistem dapat melakukan validasi alamat IP pengirim *request* berdasarkan satu atau beberapa alamat IP yang dikonfigurasi pada kolom *ip_address* dengan pemisah koma
7. Sistem harus menyediakan fitur pengelolaan akun administrator
8. Sistem harus menyediakan fitur verifikasi OTP dan *reset password* melalui email

Kebutuhan Non-Fungsional

Kebutuhan non fungsional menjelaskan kualitas sistem meliputi:

1. Keamanan: sistem harus mampu membatasi akses API hanya kepada pengguna yang memiliki token valid, aktif, dan belum kedaluwarsa
2. Konsistensi: sistem harus memberikan respons JSON yang konsisten sesuai spesifikasi *REST API* pada seluruh *endpoint*
3. Ketersediaan: sistem harus dapat diakses selama layanan berjalan di lingkungan lokal maupun jaringan internal perusahaan

3.2.2 Perancangan Sistem

Arsitektur Sistem

Sistem dirancang menggunakan arsitektur RESTful API yang memisahkan tanggung jawab antara penyedia layanan (*backend*) dan konsumen layanan (*client*). Setiap *request* yang masuk tidak langsung diproses oleh *controller*, melainkan harus melewati *middleware* keamanan terlebih dahulu. *Middleware* bertugas memeriksa token akses, alamat IP pengguna, dan masa berlaku token secara berurutan. Apabila seluruh validasi berhasil, *request* diteruskan ke *controller* lalu ke service layer. Apabila salah satu validasi gagal, sistem mengembalikan respons kesalahan dengan kode HTTP yang sesuai.

Sistem menggunakan empat koneksi basis data yang dikonfigurasi secara terpisah dalam Laravel. Basis data SQLite digunakan untuk menyimpan data kredensial pengguna *dashboard* dan token *API Key* pada tabel api_accesses. Basis data PM menggunakan driver MariaDB untuk menyimpan data proyek dari subsistem PM. Basis data Protrack menggunakan driver MariaDB untuk data proyek dan klien dari subsistem Protrack. Basis data GES menggunakan driver MySQL untuk data karyawan dari subsistem Ganesha Employee System. Pemisahan koneksi ini memastikan setiap subsistem dapat berkembang secara independen tanpa saling memengaruhi.

Perancangan *Endpoint REST API*

Daftar rute endpoint API yang dikembangkan dikelompokkan menggunakan fitur routing bawaan *Laravel* dengan menerapkan struktur prefix versi (v1) dan nama modul terkait. Spesifikasi rute *REST API*, yang mencakup metode, endpoint, dan fungsi setiap rute, disajikan pada Tabel 3.2, yang merangkum daftar endpoint, metode HTTP, fungsi, serta sumber data yang digunakan pada masing-masing layanan API.

Tabel 3.2 Perancangan *Endpoint REST API*

No	Method	*Endpoint*	Keterangan
1	GET	/api/v1/healthcheck/general	Pemeriksaan status layanan API
2	GET	/api/v1/pm/projects	Daftar proyek dari subsistem PM
3	GET	/api/v1/pm/projects/{id}	Detail proyek PM berdasarkan ID
4	GET	/api/v1/pm/projects/by-name/{name}	Detail proyek PM berdasarkan nama
5	GET	/api/v1/pm/projects/search-base64/{base64}	Pencarian proyek dengan nama ter-encode Base64
6	GET	/api/v1/pm/projects/{id}/tasks	Daftar task dalam proyek PM

7	GET	/api/v1/protrack/projects	Daftar proyek dari subsistem Protrack
8	GET	/api/v1/protrack/projects/{id}	Detail proyek berdasarkan ID
9	GET	/api/v1/protrack/*client*s	Daftar klien dari subsistem Protrack
10	GET	/api/v1/protrack/*client*s/{id}	Detail klien berdasarkan ID
11	GET	/api/v1/protrack/*user*s	Daftar pengguna dari subsistem Protrack
12	GET	/api/v1/protrack/*user*s/{id}	Detail pengguna berdasarkan ID
13	GET	/api/v1/ges/employees	Daftar karyawan dari subsistem GES
14	GET	/api/v1/ges/employees/{id}	Detail karyawan berdasarkan ID

Perancangan Struktur Basis Data SQLite

Untuk mengelola fungsionalitas *API Key*, dirancang tabel api_accesses pada basis data SQLite. Kolom *ip_address* dapat menyimpan satu atau beberapa alamat IP yang dipisahkan koma. Apabila nilainya NULL, sistem mengizinkan akses dari alamat IP mana pun. Apabila terisi, sistem hanya mengizinkan akses dari IP yang terdaftar. Struktur tabel disajikan pada Tabel 3.3, memperlihatkan atribut yang digunakan untuk menyimpan informasi API Key, alamat IP yang diizinkan, serta masa berlaku *token*.

Tabel 3.3 Struktur Tabel api_accesses pada SQLite

No	Kolom	Tipe Data	Keterangan
1	id	INTEGER	Primary key, auto increment
2	token	VARCHAR	Token unik 8 karakter, dibangkitkan otomatis saat pembuatan
3	*ip_address*	VARCHAR	Satu atau beberapa IP diizinkan, dipisah koma (NULL = semua IP)
4	expires_at	DATETIME	Batas masa berlaku token
5	created_at	DATETIME	Waktu pembuatan token
6	updated_at	DATETIME	Waktu pembaruan token terakhir

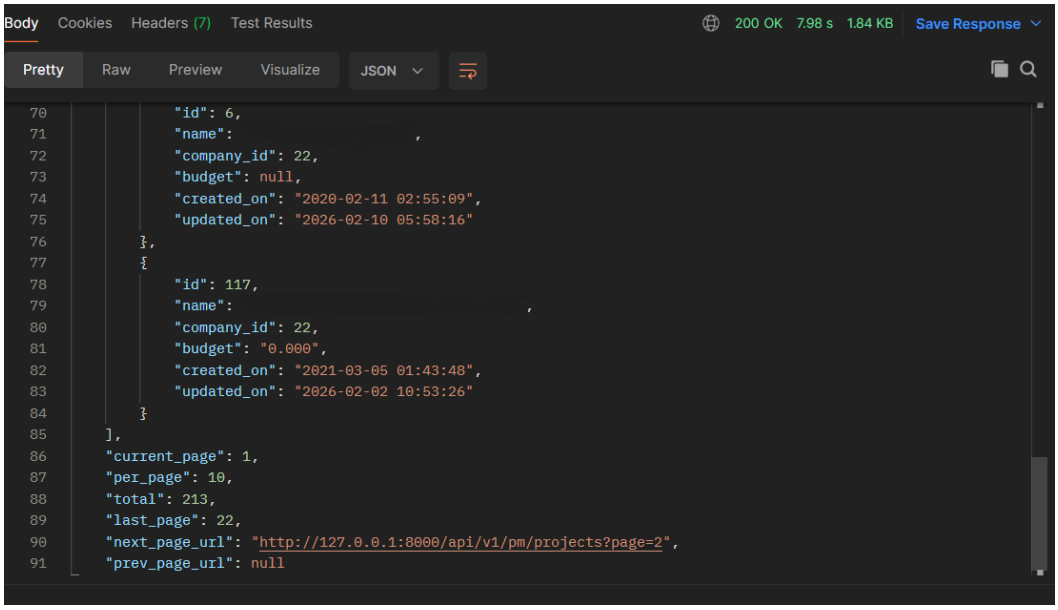
3.3 Implementasi Sistem

3.3.1 Implementasi *REST API* Data Proyek

REST API dikembangkan menggunakan *framework* Laravel untuk menyediakan layanan pertukaran data dari tiga subsistem internal perusahaan yaitu PM, Protrack, dan GES. Seluruh *endpoint* menggunakan metode HTTP GET dan mengembalikan data dalam format JSON yang terstruktur. Sistem mendukung fitur pagination untuk membatasi jumlah data yang ditampilkan pada setiap halaman.

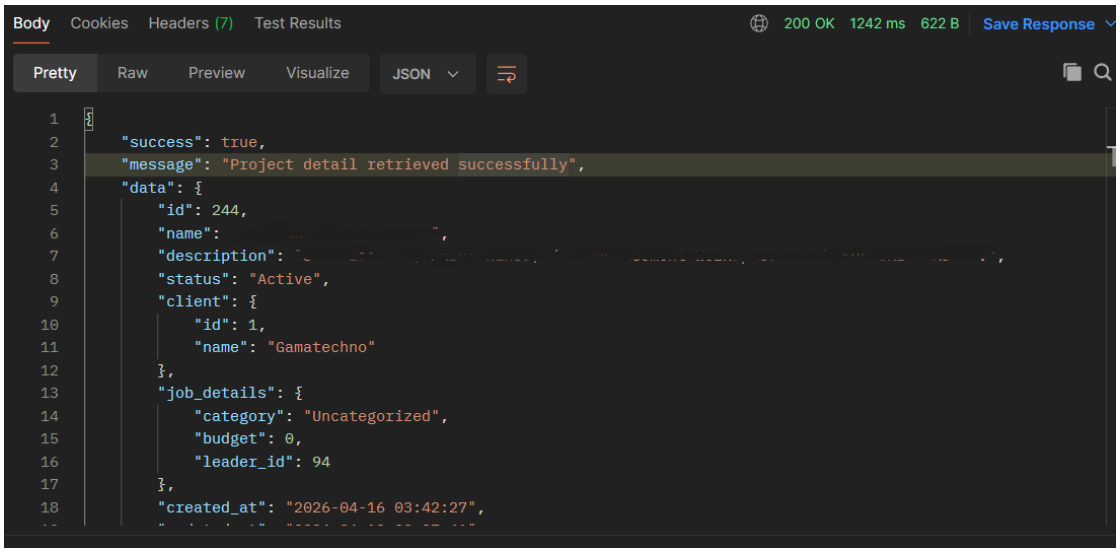
Arsitektur integrasi yang diterapkan menggunakan pendekatan multiple *database* connection. Laravel berperan sebagai lapisan integrasi yang menghubungkan beberapa sumber data berbeda ke dalam satu layanan *REST API*. Data proyek diperoleh dari basis data PM, data klien dan proyek operasional diperoleh dari basis data Protrack, sedangkan data pegawai diperoleh dari basis data GES. Setiap model Eloquent pada sistem ini dikonfigurasi untuk menggunakan koneksi basis data yang sesuai melalui properti $connection, sehingga setiap query secara otomatis diarahkan ke sumber data yang tepat tanpa memerlukan konfigurasi tambahan pada level *controller*. Dengan pendekatan ini, pengguna hanya perlu mengakses satu layanan API tanpa harus mengetahui lokasi maupun struktur basis data dari masing-masing subsistem.

Hasil implementasi dapat dilihat dari respons yang dikembalikan oleh *endpoint* GET /api/v1/pm/projects. *Endpoint* ini mengembalikan daftar proyek dalam format JSON yang dilengkapi dengan informasi pagination, sebagaimana ditunjukkan pada Gambar 3.4.



Gambar 3.4 Respons *Endpoint* GET /api/v1/pm/projects dengan Informasi Pagination

Sementara itu, *endpoint* detail seperti GET /api/v1/pm/projects/{id} mengembalikan struktur respons yang berbeda karena hanya menampilkan satu data tanpa informasi pagination, dilengkapi dengan relasi data klien dan kategori proyek. Hasil implementasi *endpoint* detail proyek ditunjukkan pada Gambar 3.5.



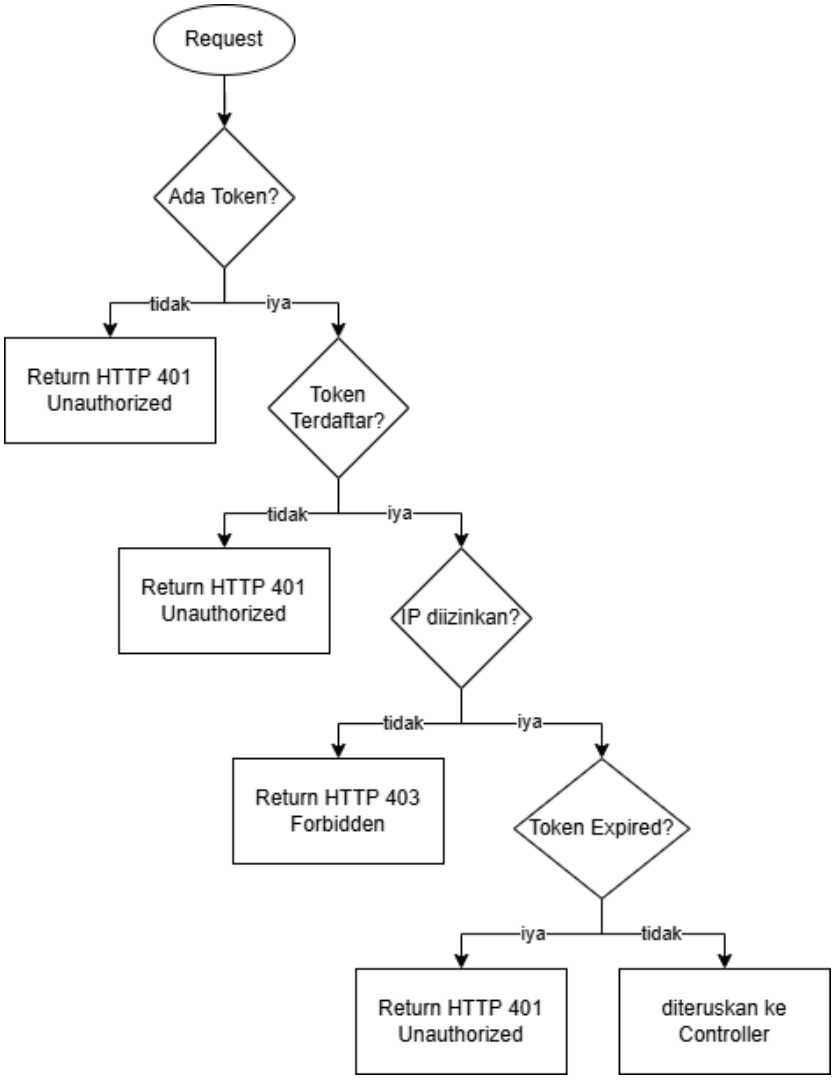
Gambar 3.5 Respons *Endpoint* GET /api/v1/pm/projects/{id} dengan Relasi Data Klien

Berdasarkan hasil implementasi yang ditunjukkan pada Gambar 3.4 dan Gambar 3.5, seluruh *endpoint* dari ketiga subsistem berhasil mengembalikan data dalam format JSON sesuai dengan struktur yang telah dirancang. *Endpoint* list mengembalikan data beserta informasi pagination, sedangkan *endpoint* detail mengembalikan satu data lengkap beserta relasi yang relevan. Pendekatan multiple *database* connection yang diterapkan terbukti berfungsi dengan baik, di mana setiap query secara otomatis diarahkan ke basis data yang sesuai tanpa memerlukan penanganan tambahan pada level *controller*.

3.3.2 Implementasi *Middleware* Validasi Token

Middleware validasi token dikembangkan sebagai komponen utama dalam sistem keamanan *REST API*. *Middleware* ini ditempatkan di antara *client* dan *controller* sehingga setiap *request* yang masuk harus melewati proses validasi terlebih dahulu sebelum data dapat diakses. Dengan mekanisme tersebut, sistem dapat memastikan bahwa hanya pengguna atau aplikasi yang memiliki token yang sah yang dapat menggunakan layanan *REST API*.

Alur proses validasi yang dilakukan oleh *middleware CheckApiToken* ditunjukkan pada Gambar 3.6. yang menggambarkan tahapan pemeriksaan mulai dari validasi keberadaan token, pencocokan token pada basis data, validasi alamat IP, hingga pemeriksaan masa berlaku token sebelum *request* diteruskan ke *controller*.



Gambar 3.6 Diagram *Middleware* Validasi Token

Tahap pertama adalah pemeriksaan keberadaan token pada *request*. Sistem membaca token yang dikirimkan, baik melalui *header Authorization* sebagai Bearer Token, *header* X-API-TOKEN, maupun *query* `string?token=`. Apabila token tidak ditemukan dari ketiga sumber tersebut, *request* langsung dihentikan dan sistem mengembalikan respons HTTP 401 *Unauthorized*.

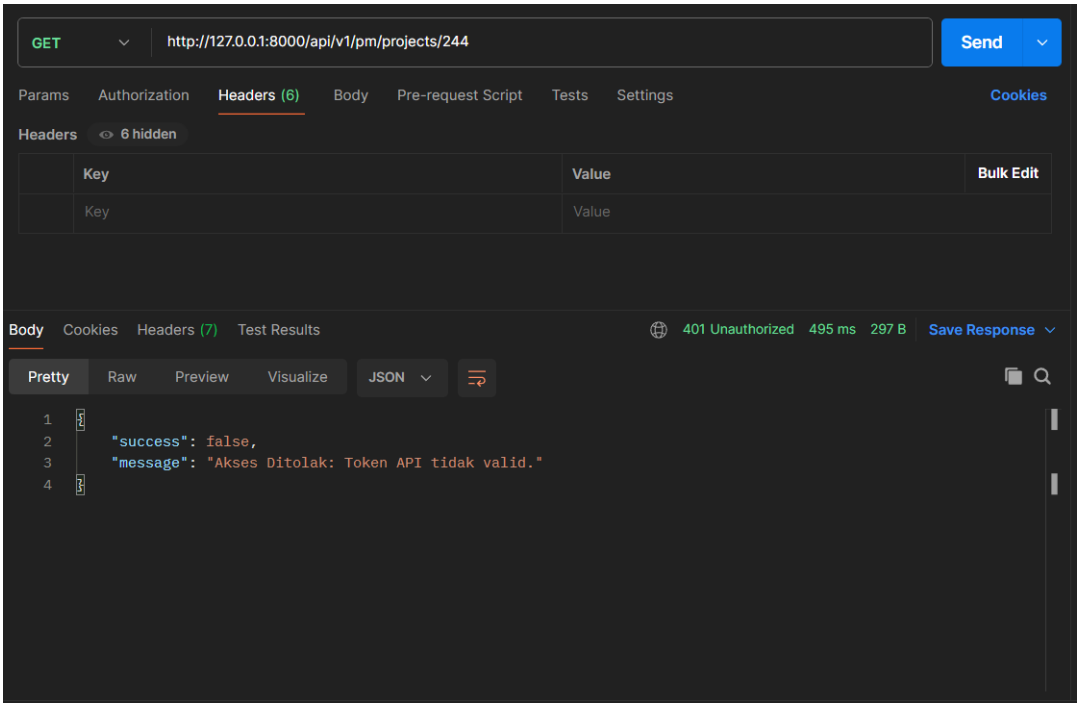
Tahap kedua adalah validasi token terhadap data yang tersimpan pada tabel api_accesses. *Middleware* melakukan pencarian token pada basis data SQLite untuk memastikan bahwa token yang digunakan benar-benar terdaftar pada sistem. Jika token tidak ditemukan, sistem mengembalikan respons HTTP 401 *Unauthorized*.

Tahap ketiga adalah validasi alamat IP. *Middleware* membaca alamat IP pengirim *request* kemudian membandingkannya dengan daftar alamat IP yang tersimpan pada kolom *ip_address*. Sistem mendukung konfigurasi lebih dari satu alamat IP dengan pemisah menggunakan tanda koma. Jika alamat IP pengirim tidak ditemukan dalam daftar yang diizinkan, sistem akan mengembalikan respons HTTP 403 Forbidden.

Tahap keempat adalah pemeriksaan masa berlaku token. Sistem membandingkan nilai expires_at dengan waktu server saat ini menggunakan library Carbon. Apabila token telah melewati batas masa berlaku yang ditentukan, akses akan ditolak dan sistem mengembalikan pesan bahwa token telah kedaluwarsa.

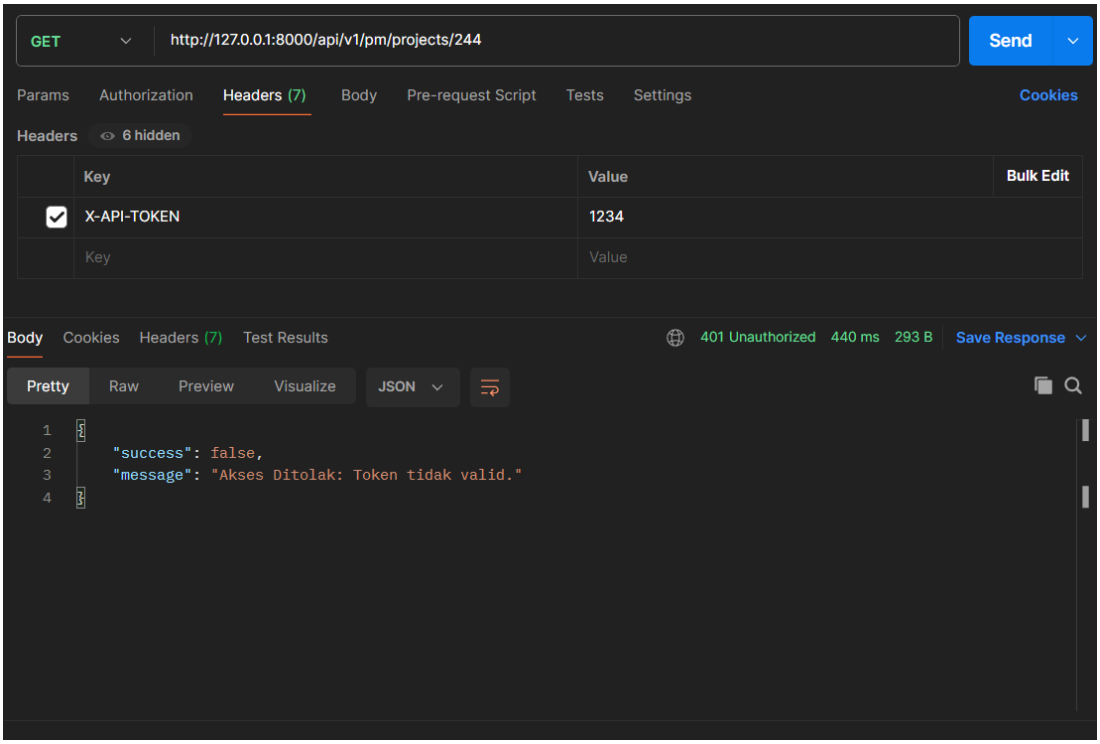
Setelah seluruh proses validasi berhasil dilakukan, *middleware* meneruskan *request* ke *controller* menggunakan mekanisme next() milik Laravel. *Controller* kemudian menjalankan proses pengambilan data dari basis data PM, Protrack, atau GES sesuai *endpoint* yang diakses.

Hasil pengujian terhadap mekanisme keamanan ini menunjukkan bahwa *middleware* bekerja sesuai dengan rancangan. *Respons* sistem ketika *request* dikirim tanpa menyertakan token ditunjukkan pada Gambar 3.7. yang memperlihatkan bahwa sistem mengembalikan kode HTTP 401 (Unauthorized) sebagai bentuk penolakan akses.



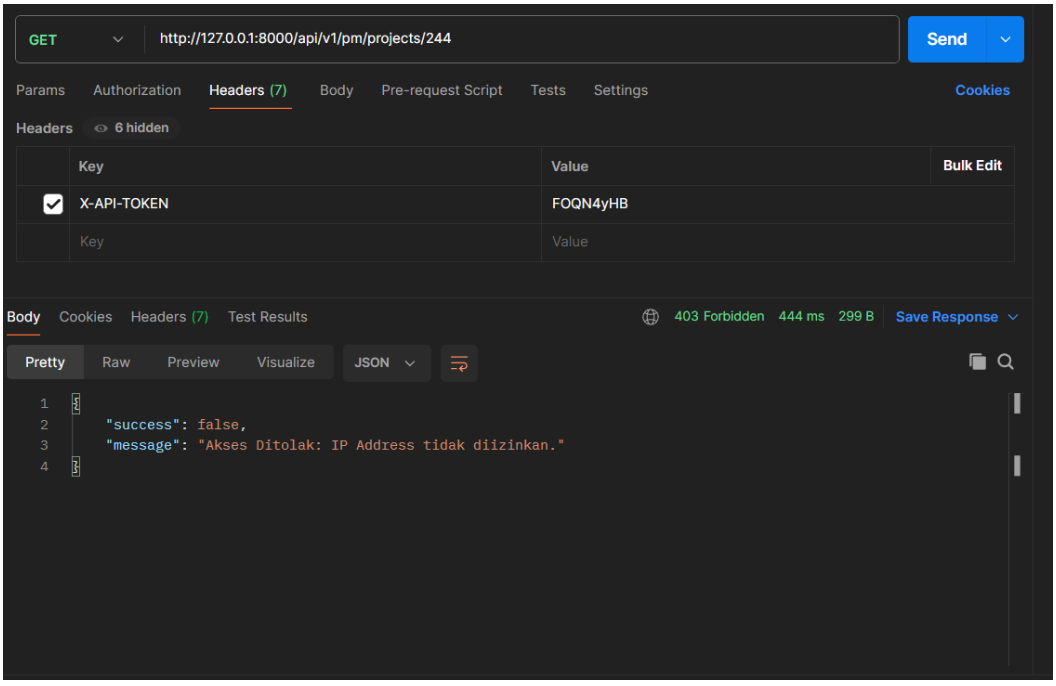
Gambar 3.7 Token API Tidak Valid

Respons sistem ketika token yang dikirim tidak terdaftar pada basis data ditunjukkan pada Gambar 3.8, yang memperlihatkan bahwa *middleware* mengembalikan kode HTTP 401 (Unauthorized) karena token tidak valid.



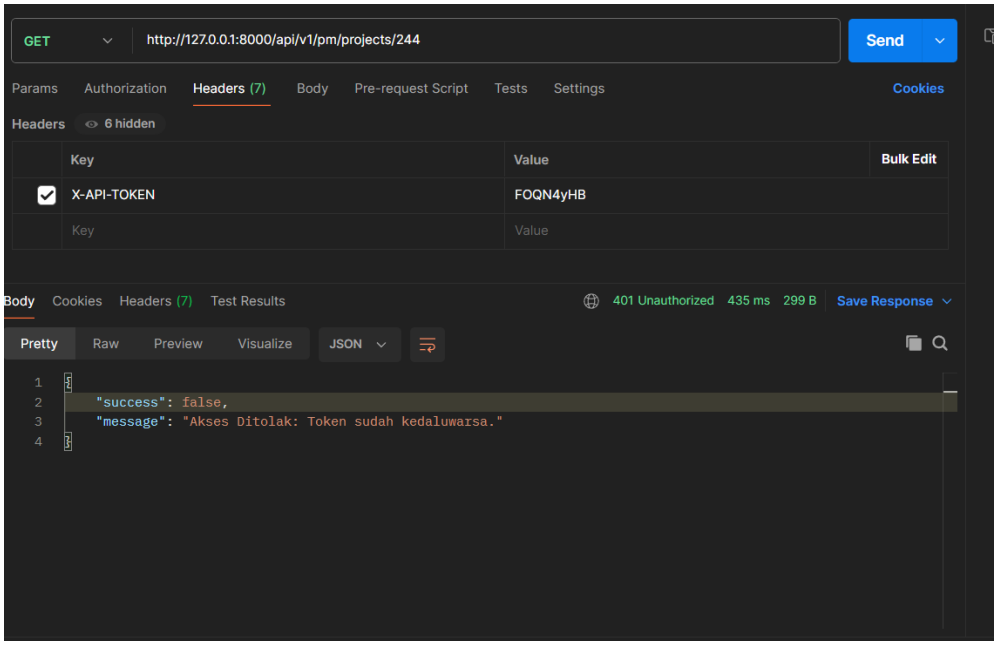
Gambar 3.8 Token API Tidak Valid

Respons sistem ketika *request* dikirim dari alamat IP yang tidak termasuk dalam daftar IP yang diizinkan ditunjukkan pada Gambar 3.9. Memperlihatkan bahwa sistem mengembalikan kode HTTP 403 (Forbidden) meskipun token yang digunakan masih valid.



Gambar 3.9 IP Tidak Diizinkan

Respons sistem ketika token telah melewati masa berlakunya ditunjukkan pada Gambar 3.10. Memperlihatkan bahwa sistem mengembalikan kode HTTP 401 (Unauthorized) karena token telah kedaluwarsa.



Gambar 3.10 Token Kedaluwarsa

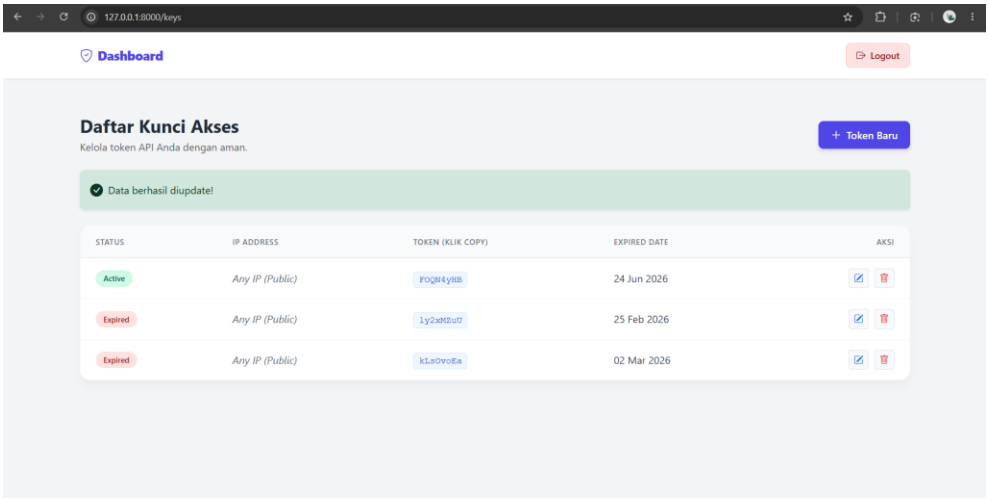
Hasil ini membuktikan bahwa *middleware* berhasil menjalankan keempat tahap validasi dengan benar dan konsisten menolak setiap permintaan yang tidak memenuhi persyaratan keamanan, sebelum permintaan tersebut sempat diteruskan ke *controller*.

3.3.3 Implementasi Sistem Manajemen *API Key* dan *Admin panel*

Mempermudah pengelolaan akses terhadap layanan *REST API*, dikembangkan sebuah *dashboard* administrasi menggunakan Filament *Admin panel* yang terintegrasi dengan Laravel. *Dashboard* ini berfungsi sebagai pusat pengelolaan *API Key* yang digunakan untuk mengakses layanan *REST API*.

Melalui *dashboard* tersebut, administrator dapat melakukan operasi *Create, Read, Update,* dan *Delete* (CRUD) terhadap data *API Key* yang tersimpan pada tabel api_accesses. Pada saat pembuatan *API Key* baru, sistem secara otomatis menghasilkan token unik yang dapat digunakan oleh *client* untuk mengakses layanan API. Selain mengelola token, administrator

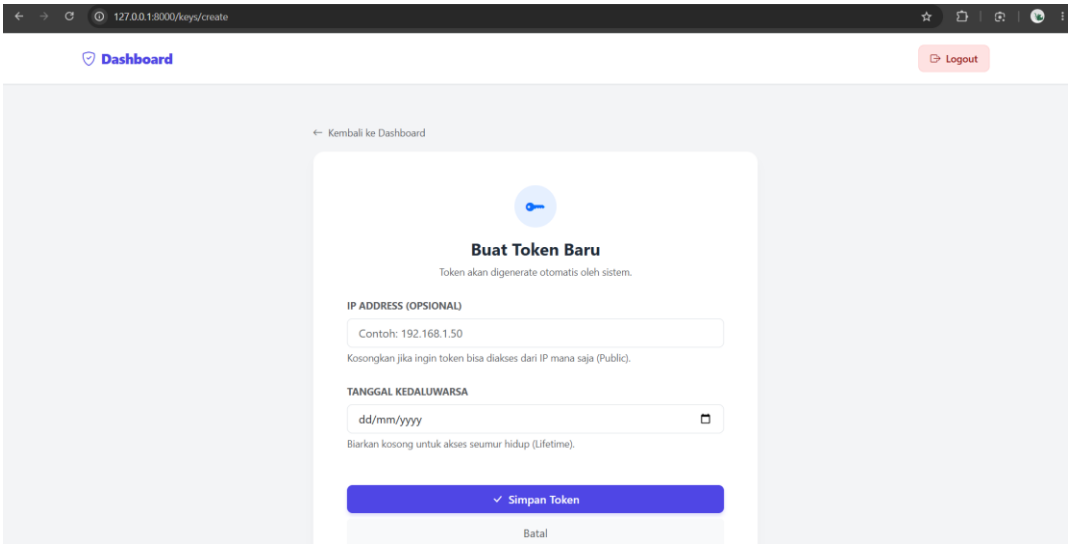
juga dapat menentukan masa berlaku token melalui atribut *expires_at*. Fitur ini memungkinkan administrator mengontrol jangka waktu penggunaan token sehingga akses dapat dibatasi sesuai kebutuhan. Tampilan *dashboard* pengelolaan *API Key* ditunjukkan pada Gambar 3.11, yang memperlihatkan fitur untuk mengelola token, mengatur alamat IP yang diizinkan, serta menentukan masa berlaku token sebagai mekanisme pengamanan akses *REST API*.



Gambar 3.11 *Dashboard* Token

Selain menghasilkan token secara otomatis, sistem juga menyediakan pengaturan masa berlaku token melalui atribut expires_at. Fitur ini memungkinkan administrator untuk membatasi periode penggunaan token sehingga akses dapat dihentikan secara otomatis setelah melewati batas waktu yang ditentukan. Administrator juga dapat menentukan alamat IP yang diizinkan menggunakan token tertentu. Dengan mekanisme tersebut, penggunaan token dapat dibatasi hanya dari perangkat atau jaringan yang telah terdaftar sehingga meningkatkan keamanan akses terhadap layanan API.

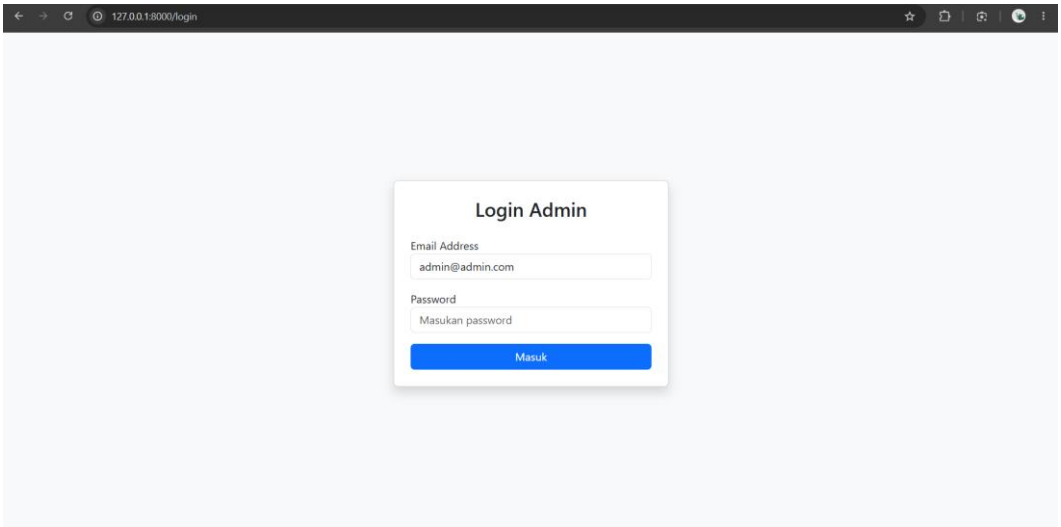
Proses pembuatan *API Key* dilakukan melalui formulir yang disediakan pada *dashboard* administrasi. Administrator dapat mengisi informasi yang diperlukan, seperti nama pengguna atau aplikasi, alamat IP yang diizinkan, masa berlaku token, serta status akses. Setelah data disimpan, sistem akan menghasilkan token secara otomatis dan menyimpannya ke dalam basis data. Tampilan formulir pembuatan *API Key* ditunjukkan pada Gambar 3.12, yang memperlihatkan halaman untuk menambahkan *API Key* baru beserta alamat IP yang diizinkan, dan masa berlaku token.



Gambar 3.12 Pembuatan Token

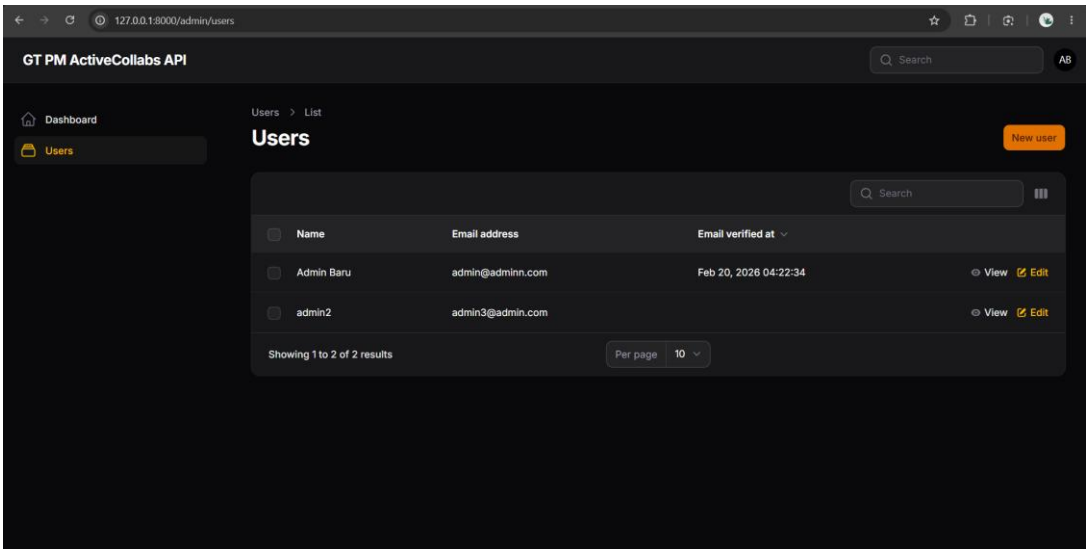
Data *API Key* yang tersimpan ditampilkan dalam bentuk tabel yang memuat informasi token, alamat IP yang diizinkan, tanggal kedaluwarsa, serta status token. Informasi tersebut memudahkan administrator dalam melakukan pemantauan dan pengelolaan akses API tanpa harus melakukan perubahan secara langsung pada basis data.

Selain fitur manajemen *API Key*, *dashboard* juga dilengkapi dengan mekanisme autentikasi pengguna untuk memastikan bahwa hanya administrator yang memiliki hak akses yang dapat melakukan pengelolaan konfigurasi sistem. Pengguna diwajibkan melakukan proses *login* sebelum dapat mengakses *dashboard* administrasi. Tampilan halaman *login* administrator ditunjukkan pada Gambar 3.13, yang memperlihatkan antarmuka autentikasi awal bagi pengguna sebelum mengakses dashboard sistem.



Gambar 3.13 Halaman *Login*

Sistem menyediakan panel khusus yang hanya dapat diakses oleh pengguna dengan peran PM (Project Manager). Melalui panel tersebut, PM dapat melakukan pengelolaan akun administrator secara penuh, meliputi penambahan pengguna baru, pengubahan informasi pengguna, serta pengaturan peran yang dimiliki. Tampilan panel pengelolaan pengguna yang hanya dapat diakses oleh PM ditunjukkan pada Gambar 3.14, yang memperlihatkan fitur untuk mengelola akun pengguna yang hanya dapat diakses oleh administrator dengan hak akses PM.



Gambar 3.14 *Dashboard User*

Pembatasan akses berbasis peran ini diterapkan untuk menjaga keamanan sistem serta memastikan bahwa fungsi administrasi pengguna hanya dapat dilakukan oleh pihak yang memiliki wewenang.

Berdasarkan hasil implementasi, *dashboard* administrasi berhasil menyediakan fasilitas pengelolaan *API Key* secara terpusat sekaligus menerapkan pembatasan hak akses berbasis peran. PM memiliki wewenang penuh termasuk pengelolaan akun, sementara pengguna lainnya hanya dapat mengakses informasi dan mengelola token sesuai batasan yang ditetapkan.

3.3.4 Implementasi Verifikasi OTP dan *Reset password*

Meningkatkan keamanan akses *dashboard* administrasi, sistem dilengkapi dengan fitur verifikasi One Time *Password* (OTP) dan *reset password* berbasis email. OTP merupakan kode verifikasi yang hanya berlaku untuk satu kali penggunaan dalam jangka waktu tertentu,

sehingga tidak dapat digunakan kembali meskipun kode tersebut diketahui oleh pihak lain. Implementasi fitur ini memanfaatkan layanan *Notification* yang disediakan oleh Laravel untuk mengirimkan kode OTP ke alamat email pengguna yang terdaftar tanpa memerlukan integrasi pihak ketiga seperti layanan SMS Gateway. Perlu ditegaskan bahwa pengiriman email pada implementasi ini menggunakan layanan email lokal bawaan Laravel, yang diuji menggunakan MailHog sebagai penampung email pada lingkungan pengembangan, bukan layanan email produksi yang terhubung ke penyedia email eksternal. Dengan pendekatan ini, sistem dapat memberikan lapisan keamanan tambahan di atas autentikasi berbasis email dan kata sandi yang sudah ada, sehingga akses tidak sah ke dashboard administrasi dapat dicegah meskipun kredensial pengguna berhasil diperoleh oleh pihak yang tidak berwenang.

Pada proses login, pengguna terlebih dahulu memasukkan alamat email dan kata sandi yang valid melalui halaman login dashboard. Setelah sistem berhasil memverifikasi kredensial pengguna, sistem tidak langsung memberikan akses ke dashboard. Pengguna terlebih dahulu diarahkan ke tahap verifikasi OTP sebagai bagian dari proses autentikasi. Selanjutnya, sistem secara otomatis memanggil fungsi pembangkitan OTP yang menghasilkan kode berupa enam digit angka acak menggunakan fungsi random_int() bawaan PHP.

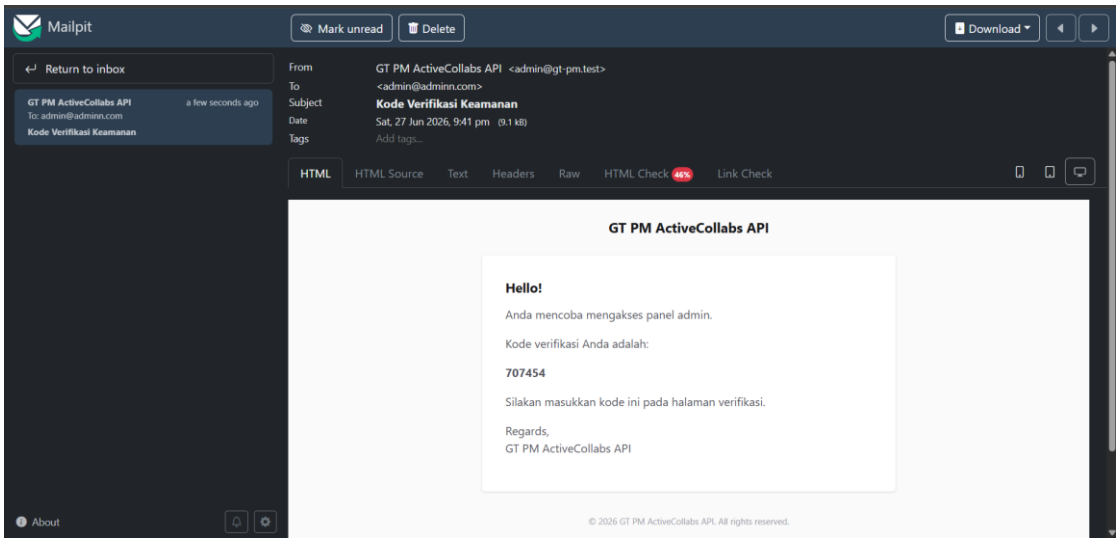
Kode OTP yang ada kemudian disimpan ke dalam session Laravel beserta dua nilai waktu, yaitu last_otp_sent_at yang mencatat waktu pengiriman OTP dalam format Unix timestamp dan otp_expires_at yang mencatat batas waktu kedaluwarsa OTP, ditetapkan lima menit setelah kode diterbitkan. Seluruh data ini disimpan dalam session, bukan pada basis data, karena OTP hanya relevan selama satu sesi autentikasi dan tidak memerlukan penyimpanan permanen. Setelah data tersimpan di session, sistem secara otomatis mengirimkan kode OTP ke alamat email pengguna yang sedang melakukan login melalui kelas notifikasi SendOtpVerification. Implementasi fungsi pembangkitan dan pengiriman OTP ditunjukkan pada Kode Gambar 3.15, yang memperlihatkan proses pembangkitan kode OTP enam digit secara acak, penyimpanan data OTP ke dalam session Laravel, serta pengiriman kode OTP ke alamat email pengguna menggunakan kelas notifikasi `SendOtpVerification`.

```
protected function generateAndSendOtp(): void
{
    $otp = (string) random_int(100000, 999999);
    session([
        'simulated_otp'    => $otp,
        'last_otp_sent_at' => now()->timestamp,
        'otp_expires_at'   => now()->addMinutes(5)->timestamp,
```

```
        ]);
        Auth::user()?->notify(new SendOtpVerification($otp));
    }
```

Gambar 3.15 Kode Program Pengiriman OTP

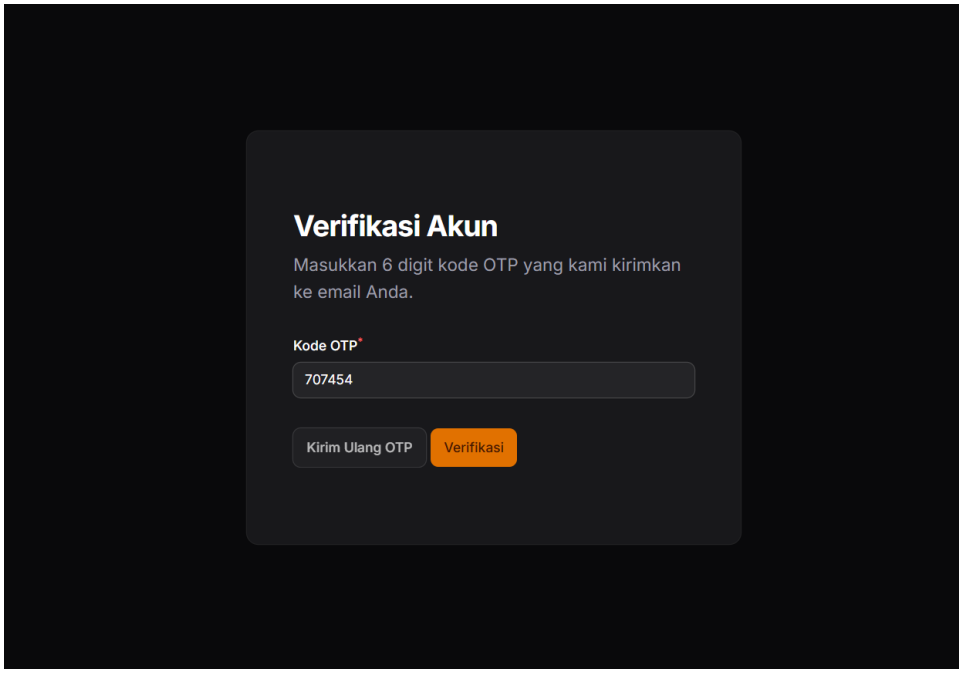
Kelas `SendOtpVerification` merupakan kelas notifikasi Laravel yang bertanggung jawab menyusun dan mengirimkan email kepada pengguna. Di dalamnya, kode OTP disisipkan ke dalam template pesan email yang berisi instruksi penggunaan kode beserta informasi masa berlakunya. Dengan memanfaatkan sistem notifikasi Laravel, pengiriman email dapat dilakukan tanpa perlu mengonfigurasi koneksi SMTP secara manual di dalam kode program seluruh konfigurasi pengiriman cukup dilakukan melalui berkas .env pada tingkat konfigurasi aplikasi. Hasil pengujian menunjukkan bahwa email OTP berhasil diterima oleh pengguna pada layanan email lokal (MailHog) dengan format pesan sebagaimana ditunjukkan pada Gambar 3.16, yang memperlihatkan pesan berisi kode OTP enam digit yang dikirimkan sistem sebagai bagian dari proses verifikasi identitas pengguna.



Gambar 3.16 Kode Email OTP

Setelah proses autentikasi email dan kata sandi berhasil dan kode OTP berhasil dikirimkan, sistem secara otomatis mengarahkan pengguna ke halaman verifikasi OTP. Pada halaman tersebut, pengguna diminta untuk memasukkan kode enam digit yang telah dikirimkan ke alamat emailnya. Pengguna tidak dapat melewati halaman ini dan langsung mengakses *dashboard* seluruh rute menuju halaman *dashboard* dilindungi oleh pemeriksaan status verifikasi OTP. Tampilan halaman verifikasi OTP ditunjukkan pada Gambar 3.17.

Memperlihatkan antarmuka bagi pengguna untuk memasukkan kode OTP enam digit yang telah dikirimkan ke alamat email sebelum sistem memberikan akses ke dashboard.



Gambar 3.17 Verifikasi Akun dengan OTP

Sistem menetapkan masa berlaku OTP selama lima menit sejak kode dikirim. Apabila pengguna tidak memasukkan kode dalam batas waktu tersebut, kode OTP dinyatakan kedaluwarsa dan pengguna perlu meminta pengiriman ulang kode yang baru. Untuk mencegah penyalahgunaan fitur pengiriman ulang misalnya oleh pihak yang mencoba membanjiri kotak masuk email pengguna dengan permintaan OTP berulang sistem menerapkan mekanisme cooldown selama 60 detik.

Mekanisme cooldown bekerja dengan cara membandingkan waktu permintaan pengiriman ulang yang baru dengan nilai `last_otp_sent_at` yang tersimpan pada *session*. Apabila selisih waktu antara keduanya belum mencapai 60 detik, sistem menolak permintaan dan menampilkan notifikasi yang memberitahukan sisa waktu tunggu kepada pengguna. Sebaliknya, apabila selisih waktu sudah melampaui 60 detik, sistem mengizinkan pengiriman ulang dan memperbarui nilai `last_otp_sent_at` pada *session* dengan waktu pengiriman yang baru. Implementasi mekanisme cooldown ini ditunjukkan pada Gambar 3.18, memperlihatkan logika sistem dalam membatasi pengiriman ulang OTP selama 60 detik untuk mencegah penyalahgunaan fitur *resend* OTP.

```
public function resend(): void
{
    $secondsRemaining = 60 - (now()->timestamp - session('last_otp_sent_at', 0));

    if ($secondsRemaining > 0) {
        $this->cooldown = $secondsRemaining;

        Notification::make()
            ->title("Tunggu {$secondsRemaining} detik sebelum kirim ulang.")
            ->danger()
            ->send();

        return;
    }

    $this->generateAndSendOtp();
    $this->cooldown = 60;

    Notification::make()
        ->title('OTP berhasil dikirim ulang')
        ->success()
        ->send();
}
```

Gambar 3.18 Kode Program Mekanisme Cooldown Pengiriman Ulang OTP

Ketika pengguna memasukkan kode OTP pada halaman verifikasi, sistem melakukan dua tahap pemeriksaan secara berurutan sebelum memberikan keputusan apakah verifikasi berhasil atau gagal.

Tahap pertama adalah pemeriksaan masa berlaku. Sistem mengambil nilai otp_expires_at dari *session* dan membandingkannya dengan waktu server saat ini menggunakan *Unix timestamp*. Apabila waktu saat ini telah melampaui nilai otp_expires_at, sistem langsung menolak verifikasi tanpa memeriksa kecocokan kode, karena kode yang sudah kedaluwarsa tidak boleh diterima meskipun nilainya benar.

Tahap kedua adalah pemeriksaan kecocokan kode. Sistem membandingkan kode yang dimasukkan pengguna dengan nilai simulated_otp yang tersimpan pada *session*. Pembandingan dilakukan secara ketat menggunakan pengecekan tipe data untuk memastikan tidak terjadi kesalahan akibat perbedaan tipe antara string dan integer. Implementasi kedua tahap validasi ini ditunjukkan pada Gambar 3.19, memperlihatkan tahapan pemeriksaan masa berlaku OTP dan pencocokan kode OTP yang dimasukkan pengguna dengan data yang tersimpan pada *session* sebelum proses autentikasi dinyatakan berhasil.

```
public function verify(): void
{
    $inputOtp   = $this->data['otp'] ?? null;
    $sessionOtp = session('simulated_otp');

    if (now()->timestamp > session('otp_expires_at', 0)) {
        Notification::make()
            ->title('OTP sudah kedaluwarsa')
            ->body('Silakan kirim ulang kode OTP.')
            ->danger()
            ->send();

        return;
    }

    if ($inputOtp === $sessionOtp) {
        session(['is_verified_session' => true]);
        session()->forget(['simulated_otp', 'last_otp_sent_at', 'otp_expires_at']);

        Notification::make()
            ->title('Verifikasi berhasil')
            ->success()
            ->send();

        redirect()->to('/admin');

        return;
    }

    Notification::make()
        ->title('Kode OTP salah')
        ->body('Kode yang Anda masukkan tidak cocok.')
        ->danger()
        ->send();
}
```

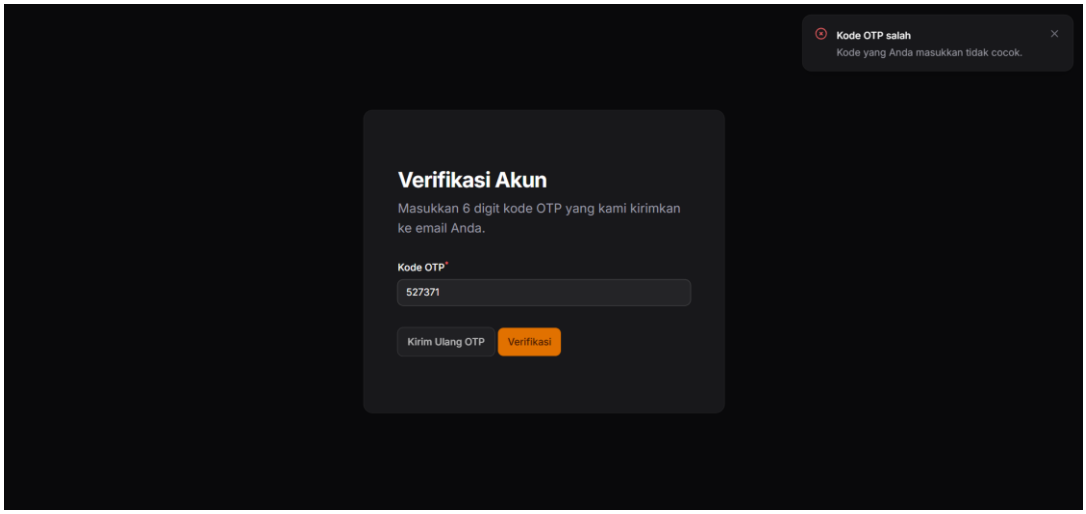
Gambar 3.19 Kode Program Logika Validasi Kode OTP

Apabila kedua pemeriksaan berhasil dilewati, sistem menampilkan notifikasi keberhasilan dan mengarahkan pengguna ke halaman *dashboard* utama. Pada saat yang sama, seluruh data OTP yang tersimpan pada *session*, meliputi simulated_otp, last_otp_sent_at, dan otp_expires_at, dihapus dari *session* untuk mencegah penggunaan ulang kode yang sama pada proses verifikasi berikutnya. Sebaliknya, apabila kode yang dimasukkan tidak sesuai atau telah kedaluwarsa, sistem menampilkan notifikasi kesalahan tanpa menghapus data *session* sehingga pengguna masih dapat mencoba kembali atau meminta pengiriman ulang kode. Tampilan

notifikasi kesalahan saat kode OTP yang dimasukkan tidak valid ditunjukkan pada Gambar 3.20, memperlihatkan pesan kesalahan sebagai pemberitahuan bahwa kode OTP yang dimasukkan tidak sesuai dengan data yang tersimpan pada sistem.

Perlu dicatat bahwa variabel penyimpanan kode OTP pada implementasi ini masih menggunakan nama simulated_otp, yang merupakan penamaan dari tahap pengembangan awal ketika mekanisme pengiriman OTP belum terintegrasi dengan layanan email. Meskipun nama variabel tersebut belum diperbarui, fungsionalitas OTP saat ini telah terhubung dengan mekanisme notifikasi email Laravel melalui kelas SendOtpVerification. Pada tahap pengembangan dan pengujian, pengiriman email menggunakan MailHog sebagai *mail server* lokal sehingga email OTP dapat diperiksa pada lingkungan pengujian tanpa dikirimkan melalui layanan email produksi. Dengan demikian, kode OTP tetap dikirim oleh aplikasi, sedangkan lingkungan pengiriman emailnya masih bersifat lokal. Penggunaan MailHog tersebut ditunjukkan pada pengujian skenario 17 hingga 20 pada Tabel 3.4 Hasil Pengujian Sistem.

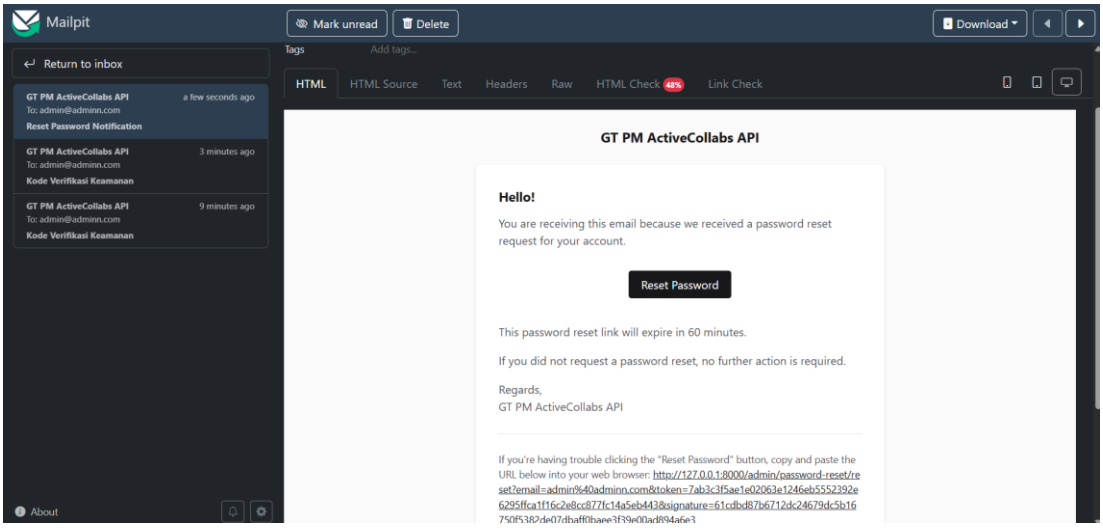
Pada implementasi ini, kode OTP beserta informasi waktu pengiriman dan masa berlaku disimpan pada *session* server sebagai data sementara selama proses verifikasi. Penggunaan *session* menyebabkan keberlangsungan data OTP bergantung pada ketersediaan data *session*. Apabila data *session* yang menyimpan OTP hilang atau tidak tersedia, termasuk pada kondisi tertentu setelah *server restart*, pengguna yang sedang berada pada tahap verifikasi tidak dapat melanjutkan proses menggunakan OTP sebelumnya. Dalam kondisi tersebut, pengguna perlu melakukan proses autentikasi kembali atau meminta pengiriman OTP baru. Keterbatasan penggunaan *session* sebagai media penyimpanan OTP tersebut menjadi salah satu aspek yang perlu dipertimbangkan dalam penerapan sistem pada lingkungan produksi dan dibahas lebih lanjut pada subbab Saran.



Gambar 3.20 Notifikasi Kode OTP Salah

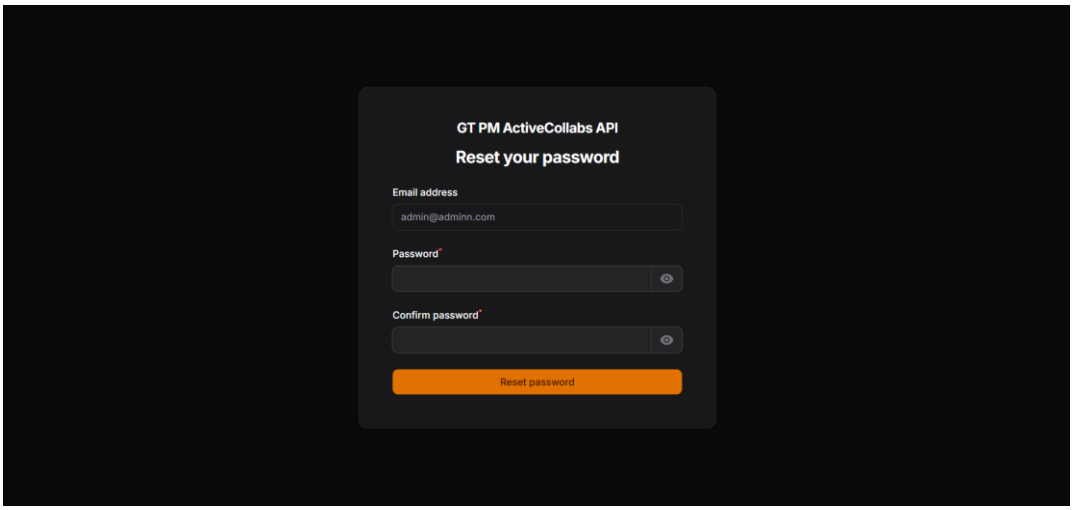
Selain digunakan pada proses *login*, mekanisme OTP yang sama juga diterapkan pada fitur *reset password*. Kedua alur ini menggunakan fungsi pembangkitan dan pengiriman OTP yang identik, namun memiliki titik masuk yang berbeda. Pada alur *login*, OTP dibangkitkan setelah autentikasi email dan kata sandi berhasil. Pada alur *reset password*, OTP dibangkitkan dari halaman tersendiri yang dapat diakses oleh pengguna yang belum berhasil melakukan *login*.

Pengguna yang lupa kata sandinya dapat mengakses halaman permintaan *reset password* melalui tautan yang tersedia pada halaman *login*. Pada halaman tersebut, pengguna diminta untuk memasukkan alamat email yang terdaftar pada sistem. Sistem kemudian memverifikasi keberadaan alamat email tersebut pada basis data pengguna apabila email tidak ditemukan, sistem menampilkan pesan kesalahan tanpa mengirimkan OTP. Apabila email ditemukan, sistem mengirimkan kode OTP menggunakan mekanisme yang sama seperti pada proses *login* dan mengirimkannya ke alamat email tersebut, melalui layanan email lokal bawaan laravel. Tampilan halaman permintaan *reset password* ditunjukkan pada Gambar 3.21, memperlihatkan halaman perminataan *reset password* telah dikirm melalui email lokal bawaan laravel.



Gambar 3.21 Tampilan Email *Reset password*

Setelah pengguna berhasil memasukkan kode OTP yang valid pada halaman verifikasi, sistem memberikan akses ke halaman pembuatan kata sandi baru. Pada halaman ini, pengguna diminta untuk memasukkan kata sandi baru beserta konfirmasinya. Sistem memvalidasi bahwa kedua nilai yang dimasukkan cocok dan memenuhi panjang minimum yang ditetapkan sebelum menyimpan perubahan. Tampilan halaman pembuatan kata sandi baru ditunjukkan pada Gambar 3.22, memperlihatkan formulir untuk memasukkan kata sandi baru beserta konfirmasinya setelah pengguna berhasil melewati proses verifikasi OTP.



Gambar 3.22 Tampilan Pembuatan Kata Sandi Baru

Kata sandi baru yang telah divalidasi kemudian disimpan ke dalam basis data dalam bentuk hash menggunakan fungsi Hash::make() yang disediakan oleh Laravel. Fungsi Hash::make()

menggunakan algoritma bcrypt secara bawaan, yang merupakan algoritma hashing satu arah artinya kata sandi yang telah di-hash tidak dapat dikembalikan ke bentuk aslinya. Dengan mekanisme ini, kata sandi asli pengguna tidak pernah tersimpan dalam bentuk teks biasa pada basis data sehingga tetap aman meskipun terjadi kebocoran data. Setelah kata sandi berhasil diperbarui, seluruh data OTP pada *session* dihapus dan pengguna diarahkan kembali ke halaman *login* untuk melakukan autentikasi menggunakan kata sandi yang baru.

Implementasi OTP berbasis *session* ini dipilih karena proses verifikasi hanya dilakukan selama satu sesi autentikasi dan tidak memerlukan penyimpanan permanen pada basis data. Dibandingkan dengan pendekatan penyimpanan OTP di basis data yang memerlukan pembuatan tabel baru beserta proses pembersihan data yang kedaluwarsa secara berkala, pendekatan berbasis *session* ini lebih ringan dan lebih mudah dikelola. Meskipun demikian, kode OTP hanya berlaku selama lima menit, tidak dapat digunakan lebih dari satu kali, dan hanya dapat diakses dalam sesi yang sama dengan sesi yang membangkitkannya. Perlu digaris bawahi bahwa penyimpanan OTP pada *session* server ini memiliki keterbatasan, yaitu seluruh kode OTP yang sedang aktif akan hilang apabila server mengalami restart, sehingga pengguna yang sedang berada pada tahap verifikasi harus mengulang proses login dari awal. Keterbatasan ini dibahas lebih lanjut pada subbab Saran.

3.4 Pengujian Sistem

Seluruh *endpoint REST API* yang telah dibangun diuji secara fungsional menggunakan Postman. Pengujian difokuskan untuk memvalidasi fungsionalitas middleware pengaman, keakuratan respons data JSON, serta fungsionalitas fitur autentikasi pengguna. Hasil pengujian disajikan pada Tabel 3.4.

Tabel 3.4 Hasil Pengujian Sistem

No	Skenario Pengujian	Input	Expected Output	Actual Output	Status
1	*Request* dengan token valid dan aktif	*Header* X-API-TOKEN: token aktif	HTTP 200, data JSON	HTTP 200, data JSON	Berhasil
2	*Request* tanpa token	Tidak ada *header* X-API-TOKEN	HTTP 401 Unauthorized	HTTP 401 Unauthorized	Berhasil
3	*Request* dengan token tidak terdaftar	*Header* X-API-TOKEN: token acak	HTTP 401 Unauthorized	HTTP 401 Unauthorized	Berhasil

4	*Request* dengan token kedaluwarsa	Token dengan expires_at terlewati	Pesan token kedaluwarsa	Pesan token kedaluwarsa	Berhasil
5	*Request* dari IP tidak diizinkan	Token valid, IP tidak sesuai whitelist	HTTP 403 Forbidden	HTTP 403 Forbidden	Berhasil
6	GET /api/v1/pm/projects	Token valid	HTTP 200, daftar proyek PM	HTTP 200, daftar proyek PM	Berhasil
7	GET /api/v1/pm/projects/{id}	Token valid, ID valid	HTTP 200, detail proyek	HTTP 200, detail proyek	Berhasil
8	GET /api/v1/pm/projects/{id}/tasks	Token valid, ID proyek 211	HTTP 200, daftar task	HTTP 200, daftar task	Berhasil
9	GET /api/v1/protrack/projects	Token valid	HTTP 200, daftar proyek Protrack	HTTP 200, daftar proyek Protrack	Berhasil
10	GET /api/v1/protrack/*client*s	Token valid	HTTP 200, daftar klien	HTTP 200, daftar klien	Berhasil
11	GET /api/v1/protrack/*user*s	Token valid	HTTP 200, daftar pengguna	HTTP 200, daftar pengguna	Berhasil
12	GET /api/v1/ges/employees	Token valid	HTTP 200, daftar karyawan	HTTP 200, daftar karyawan	Berhasil
13	GET /api/v1/healthcheck/general	Token valid	HTTP 200, status layanan aktif	HTTP 200, status layanan aktif	Berhasil
14	Membuat *API Key* baru	Data token diisi lengkap	Token baru tersimpan di daftar	Token baru tersimpan di daftar	Berhasil
15	Mengedit konfigurasi *API Key*	Ubah tanggal kedaluwarsa	Data berhasil diperbarui	Data berhasil diperbarui	Berhasil
16	Menghapus *API Key*	Pilih token, klik hapus	Token terhapus dari daftar	Token terhapus dari daftar	Berhasil
17	Pengiriman OTP *Login*	Email pengguna valid	Kode OTP berhasil dikirim ke email	Kode OTP berhasil dikirim ke email	Berhasil
18	Verifikasi OTP *Login* Valid	Kode OTP sesuai	*Login* berhasil	*Login* berhasil	Berhasil
19	Verifikasi OTP *Login* Tidak Valid	Kode OTP salah	Pesan OTP tidak valid	Pesan OTP tidak valid	Berhasil
20	*Reset password* Menggunakan OTP	OTP valid dan *password* baru	*Password* berhasil diperbarui	*Password* berhasil diperbarui	Berhasil

Perlu ditegaskan bahwa cakupan pengujian pada sistem ini terdiri atas pengujian fungsional (*black box*) menggunakan Postman dan pengujian keamanan terbatas terhadap

mekanisme keamanan yang diterapkan pada sistem. Pengujian keamanan yang dilakukan mencakup validasi pembatasan akses berdasarkan alamat IP yang tidak diizinkan. Sementara itu, pengujian keamanan lain yang relevan untuk sistem bertema keamanan, seperti uji *SQL injection*, uji *brute force* atau *token guessing*, pengujian bypass terhadap mekanisme keamanan lainnya, serta pengujian beban atau performa, belum dilakukan dalam sistem ini. Keterbatasan cakupan pengujian tersebut menjadi salah satu pertimbangan untuk pengembangan dan pengujian lebih lanjut

BAB IV
REFLEKSI PELAKSANAAN MAGANG

4.1 Relevansi Akademik

Berdasarkan pengalaman yang diperoleh selama melaksanakan kegiatan magang di PT Gamatechno, terdapat beberapa perbedaan antara teori yang dipelajari selama perkuliahan dengan implementasi yang dilakukan dalam lingkungan kerja profesional. Perbedaan tersebut memberikan pemahaman baru mengenai bagaimana teori yang dipelajari di bangku perkuliahan diterapkan untuk memenuhi kebutuhan sistem yang digunakan secara nyata. Berikut merupakan beberapa perbandingan antara teori yang dipelajari selama perkuliahan dengan implementasi yang dilakukan selama kegiatan magang.

1. **Pengembangan *REST API***
   a. Selama perkuliahan, penulis mempelajari *REST API* sebagai metode komunikasi antara *client* dan server yang menggunakan protokol HTTP. Implementasi *REST API* yang dilakukan umumnya masih terbatas pada kebutuhan tugas perkuliahan dan proyek akademik dengan jumlah *endpoint* yang relatif sedikit serta tingkat keamanan yang sederhana.
   b. Selama kegiatan magang, *REST API* menjadi komponen utama yang digunakan dalam proses pertukaran data antar sistem. Pengembangan API tidak hanya berfokus pada pembuatan *endpoint*, tetapi juga memperhatikan validasi data, pengelolaan respons, dokumentasi API, keamanan akses, serta integrasi dengan sistem lain yang telah digunakan perusahaan. Pengalaman ini memberikan pemahaman bahwa pengembangan *REST API* pada lingkungan industri memiliki tingkat kompleksitas yang lebih tinggi dibandingkan implementasi yang dilakukan selama perkuliahan.

2. **Penggunaan Laravel *Framework***
   a. Pada saat perkuliahan, Laravel digunakan sebagai *framework* untuk membangun aplikasi web dengan memanfaatkan fitur-fitur dasar seperti routing, *controller*, model, migration, dan autentikasi sederhana. Fokus penggunaan Laravel lebih banyak diarahkan pada penyelesaian tugas dan proyek akademik.

b. Selama kegiatan magang, Laravel digunakan sebagai *framework* utama dalam pengembangan *backend* sistem. Penulis terlibat dalam implementasi *controller*, *middleware*, validasi *request*, pengelolaan *API Key*, integrasi basis data, serta pengembangan berbagai *endpoint REST API* yang digunakan oleh aplikasi klien. Pengalaman tersebut memberikan pemahaman bahwa Laravel tidak hanya digunakan untuk membangun aplikasi web, tetapi juga mampu digunakan sebagai fondasi utama dalam pengembangan layanan *backend* yang terstruktur dan mudah dipelihara.

3. Implementasi Keamanan Sistem

a. Selama perkuliahan, konsep keamanan sistem umumnya dipelajari melalui materi autentikasi, otorisasi, dan keamanan basis data. Implementasi yang dilakukan pada proyek akademik masih relatif sederhana dan belum banyak membahas pengamanan layanan API secara mendalam.

b. Pada kegiatan magang, penulis memperoleh pengalaman langsung dalam mengimplementasikan berbagai mekanisme keamanan seperti validasi *API Key*, validasi token, OTP Verification, serta pembatasan akses berdasarkan alamat IP tertentu. Implementasi tersebut menunjukkan bahwa keamanan layanan API memerlukan beberapa lapisan perlindungan agar akses yang diberikan hanya dapat dilakukan oleh pengguna atau sistem yang memiliki hak akses yang sesuai.

4. Pengelolaan Basis Data

a. Selama perkuliahan, penulis lebih sering menggunakan MySQL sebagai basis data utama dalam pengembangan aplikasi. Pengelolaan basis data umumnya dilakukan menggunakan phpMyAdmin atau tools sejenis untuk kebutuhan proyek akademik.

b. Selama kegiatan magang, penulis tidak hanya menggunakan MySQL sebagai basis data utama sistem, tetapi juga memanfaatkan SQLite untuk kebutuhan tertentu, seperti penyimpanan data *API Key* dan konfigurasi pendukung lainnya. Penggunaan dua jenis basis data dalam satu sistem memberikan pengalaman baru mengenai pemilihan teknologi yang disesuaikan dengan kebutuhan, performa, dan efisiensi pengelolaan data.

5. Metode Pengembangan Perangkat Lunak

a. Pada proyek-proyek perkuliahan, proses pengembangan perangkat lunak sering dilakukan secara fleksibel tanpa adanya pembagian tugas yang jelas, evaluasi berkala, maupun pengelolaan backlog yang terstruktur.
b. Selama kegiatan magang, proses pengembangan dilakukan menggunakan pendekatan Agile sehingga setiap tugas memiliki target, prioritas, dan evaluasi yang jelas. Penulis memperoleh pengalaman bekerja dalam tim pengembang yang melibatkan proses diskusi, pembagian tugas, revisi fitur, serta penyelesaian permasalahan berdasarkan kebutuhan sistem yang terus berkembang. Pendekatan tersebut membuat proses pengembangan menjadi lebih terarah dan terukur.

Secara keseluruhan, kegiatan magang di PT Gamatechno memberikan kesempatan kepada penulis untuk menerapkan pengetahuan yang diperoleh selama perkuliahan ke dalam lingkungan kerja profesional. Selain meningkatkan kemampuan teknis dalam pengembangan *backend* dan keamanan *REST API*, kegiatan ini juga memberikan pengalaman mengenai pentingnya kolaborasi tim, manajemen proyek, serta penerapan standar pengembangan perangkat lunak yang digunakan di industri. Pengalaman tersebut menjadi bekal yang berharga bagi penulis dalam memasuki dunia kerja sebagai pengembang perangkat lunak.

4.2 Pembelajaran Magang

1. Manfaat

Manfaat yang diperoleh penulis selama melaksanakan kegiatan magang di PT Gamatechno adalah sebagai berikut.

a. Penulis memperoleh pemahaman yang lebih mendalam mengenai proses pengembangan *backend* menggunakan *framework* Laravel. Jika selama perkuliahan Laravel lebih banyak digunakan untuk kebutuhan akademik, selama magang penulis dapat melihat secara langsung bagaimana Laravel digunakan dalam pengembangan sistem yang digunakan pada lingkungan kerja profesional.
b. Penulis memperoleh pengalaman dalam membangun dan mengelola *REST API* yang digunakan sebagai media pertukaran data antar sistem. Melalui kegiatan tersebut, penulis memahami pentingnya struktur *endpoint* yang baik, validasi data, pengelolaan respons, serta dokumentasi API agar layanan yang dikembangkan dapat digunakan dengan mudah oleh sistem lain.

c. Penulis memperoleh pembelajaran mengenai implementasi keamanan layanan API melalui penggunaan *API Key*, validasi token, OTP Verification, dan pembatasan akses berdasarkan alamat IP. Pengalaman tersebut memberikan pemahaman bahwa aspek keamanan merupakan bagian yang sangat penting dalam pengembangan sistem dan harus diperhatikan sejak awal proses pengembangan.
d. Penulis memperoleh pengalaman menggunakan dua jenis basis data yang berbeda, yaitu MySQL dan SQLite. Pengalaman ini memberikan wawasan baru mengenai pemilihan teknologi basis data berdasarkan kebutuhan sistem dan karakteristik data yang dikelola.
e. Selain kemampuan teknis, penulis juga memperoleh pengalaman bekerja dalam lingkungan profesional, termasuk melakukan koordinasi dengan mentor, menerima masukan terhadap hasil pekerjaan, melakukan perbaikan berdasarkan evaluasi, serta menyelesaikan tugas sesuai target yang telah ditentukan.

2. Kendala, Hambatan, dan Tantangan

Selama pelaksanaan kegiatan magang, penulis juga menghadapi beberapa kendala, hambatan, dan tantangan yang memberikan pengalaman pembelajaran yang berharga.

Kendala pertama yang dirasakan adalah proses adaptasi terhadap struktur proyek yang telah dikembangkan sebelumnya. Berbeda dengan proyek perkuliahan yang umumnya dibangun dari awal, selama magang penulis harus memahami alur sistem, struktur kode, serta hubungan antar komponen yang telah dibuat oleh pengembang sebelumnya. Kondisi tersebut memerlukan waktu untuk mempelajari dokumentasi dan memahami alur kerja sistem secara menyeluruh.

Kendala berikutnya adalah memahami mekanisme keamanan yang diterapkan pada layanan API. Implementasi *API Key*, validasi token, OTP Verification, serta pembatasan akses berdasarkan alamat IP merupakan hal yang belum banyak ditemui pada proyek akademik. Untuk mengatasi kendala tersebut, penulis mempelajari dokumentasi teknis, melakukan pengujian secara mandiri, serta berdiskusi dengan mentor untuk memahami alur keamanan yang diterapkan pada sistem.

Tantangan lainnya adalah memastikan bahwa layanan API yang dikembangkan dapat berjalan dengan baik dan menghasilkan data yang sesuai dengan kebutuhan pengguna. Setiap perubahan yang dilakukan harus diuji terlebih dahulu untuk menghindari kesalahan yang dapat memengaruhi sistem secara keseluruhan. Hal ini mengajarkan penulis pentingnya proses

pengujian dan validasi sebelum suatu fitur diterapkan ke lingkungan produksi. Selain itu, penulis juga menghadapi tantangan dalam menyesuaikan diri dengan kebutuhan sistem yang dapat berubah selama proses pengembangan berlangsung. Perubahan kebutuhan tersebut mengharuskan penulis untuk mampu beradaptasi, melakukan revisi terhadap fitur yang telah dibuat, serta mencari solusi yang sesuai dengan kebutuhan pengguna dan perusahaan.

Secara keseluruhan, kegiatan magang memberikan pengalaman yang sangat berharga bagi penulis. Selain meningkatkan kemampuan teknis dalam pengembangan *backend* dan *REST API*, kegiatan magang juga memberikan pemahaman mengenai proses kerja profesional, pentingnya komunikasi dalam tim, serta kemampuan beradaptasi terhadap perubahan kebutuhan sistem. Pengalaman tersebut menjadi bekal yang sangat bermanfaat untuk menghadapi dunia kerja di bidang teknologi informasi pada masa mendatang.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil pengembangan sistem dan pelaksanaan kegiatan magang di PT Gamatechno Indonesia, seluruh target pengembangan yang telah ditetapkan dapat direalisasikan dengan baik.

Tujuan pertama, yaitu mengembangkan sistem manajemen *API Key* berbasis Laravel *Framework* yang mampu mengelola siklus token akses, telah tercapai. Sistem yang dibangun menyediakan fasilitas pembuatan, perubahan, dan penghapusan *API Key* melalui *admin panel*. Setiap *API Key* juga dilengkapi dengan pengaturan masa berlaku melalui atribut *expires_at* serta daftar alamat IP yang diizinkan melalui atribut *ip_address*. Dengan demikian, administrator dapat mengelola akses layanan API secara terpusat tanpa harus melakukan perubahan langsung pada basis data.

Tujuan kedua, yaitu menerapkan mekanisme keamanan berlapis pada layanan *REST API*, juga telah dicapai. *Middleware CheckApi*Token berhasil diimplementasikan secara global pada seluruh *endpoint* API dan menjalankan empat tahapan validasi secara berurutan, meliputi pemeriksaan keberadaan token, pencocokan token dengan data pada basis data SQLite, validasi alamat IP pengirim, serta pemeriksaan masa berlaku token. Berdasarkan hasil pengujian menggunakan Postman, setiap permintaan yang tidak memenuhi salah satu tahapan validasi tersebut dapat ditolak dengan respons HTTP yang sesuai, yaitu 401 Unauthorized dan 403 Forbidden.

Tujuan ketiga, yaitu menyediakan layanan *REST API* yang terstruktur untuk mengintegrasikan data dari tiga subsistem internal perusahaan, telah terealisasi. Sistem mampu menghubungkan basis data PM, Protrack, dan GES melalui pendekatan *multiple database connection* dengan memanfaatkan properti *$connection* pada setiap model *Eloquent*. Sebanyak 14 *endpoint REST API* berhasil dikembangkan dan menghasilkan *respons* JSON yang konsisten dengan format *success*, message, dan data, serta dilengkapi informasi *pagination* pada *endpoint* yang mengembalikan data dalam jumlah besar.

Tujuan keempat, yaitu meningkatkan keamanan autentikasi pengguna pada *admin panel* melalui mekanisme verifikasi OTP, juga telah tercapai. Implementasi OTP dilakukan menggunakan mekanisme *session* Laravel dengan kode verifikasi enam digit yang berlaku selama lima menit serta dilengkapi fitur *cooldown* selama 60 detik untuk membatasi

pengiriman ulang kode. Hasil pengujian menunjukkan bahwa akses ke *dashboard* administrasi hanya dapat diperoleh setelah pengguna berhasil melewati dua tahapan verifikasi, yaitu autentikasi menggunakan email dan kata sandi, kemudian dilanjutkan dengan verifikasi kode OTP yang dikirimkan melalui email.

Secara keseluruhan, sistem yang dikembangkan selama kegiatan magang ini mampu menjawab permasalahan yang sebelumnya dihadapi oleh PT Gamatechno Indonesia, yaitu belum tersedianya mekanisme autentikasi yang memadai pada layanan *REST API* dan belum adanya pengelolaan token akses secara terpusat. Penerapan *middleware* validasi token, manajemen *API Key* berbasis Filament, serta verifikasi OTP menghasilkan mekanisme keamanan berlapis yang lebih terstruktur, mudah dikelola, dan mendukung peningkatan keamanan layanan API perusahaan.

5.2 Saran

Meskipun seluruh target pengembangan telah berhasil diimplementasikan, masih terdapat beberapa aspek yang dapat dikembangkan lebih lanjut pada implementasi berikutnya.

Pertama, sistem belum dilengkapi dengan fitur pencatatan aktivitas penggunaan *API Key*. Setiap permintaan yang masuk ke *endpoint* belum menghasilkan rekam jejak yang menyimpan informasi mengenai identitas pengguna, alamat IP pengirim, *endpoint* yang diakses, serta waktu akses. Padahal, informasi tersebut dapat dimanfaatkan untuk mendeteksi aktivitas mencurigakan maupun penyalahgunaan token. Oleh karena itu, pengembangan selanjutnya disarankan untuk menambahkan tabel log pada basis data SQLite dan melakukan pencatatan aktivitas secara otomatis melalui *middleware CheckApi*Token untuk setiap permintaan yang diterima, baik yang berhasil diproses maupun yang ditolak.

Kedua, sistem belum menyediakan mekanisme pembatasan jumlah permintaan dalam rentang waktu tertentu (rate limiting). Kondisi ini memungkinkan token yang valid digunakan untuk mengirimkan permintaan dalam jumlah besar secara terus-menerus, yang berpotensi membebani server dan meningkatkan risiko serangan *brute force*. Oleh sebab itu, disarankan untuk menerapkan fitur *throttle* bawaan Laravel melalui konfigurasi route atau *middleware*.

Ketiga, sistem yang dikembangkan saat ini masih dijalankan pada lingkungan lokal dan jaringan internal perusahaan karena proses pengembangan belum mencakup tahap *deployment* ke server produksi. Agar sistem dapat dimanfaatkan secara optimal dalam operasional perusahaan, perlu dilakukan proses *deployment* ke lingkungan produksi dengan menerapkan konfigurasi keamanan tambahan, seperti penggunaan protokol HTTPS, penyimpanan

environment variable yang terenkripsi, serta konfigurasi firewall untuk membatasi akses ke server.

Keempat, *admin panel* yang tersedia saat ini masih berfokus pada pengelolaan *API Key* dan akun pengguna. Pada pengembangan berikutnya, *dashboard* administrasi dapat diperluas dengan menambahkan fitur pemantauan, seperti grafik penggunaan token, jumlah permintaan per hari, serta daftar token yang mendekati masa kedaluwarsa. Penambahan fitur tersebut diharapkan dapat membantu administrator dalam melakukan pengawasan dan pengelolaan akses layanan API secara lebih proaktif.

Kelima, cakupan pengujian keamanan pada aplikasi ini masih terbatas pada beberapa mekanisme keamanan yang diterapkan pada sistem, khususnya validasi pembatasan akses berdasarkan alamat IP. Pengujian keamanan lainnya, seperti pengujian *SQL injection*, pengujian *brute force* atau *token guessing*, pengujian bypass terhadap mekanisme keamanan, serta pengujian beban dan performa belum dilakukan pada aplikasi ini. Oleh karena itu, selanjutnya disarankan untuk melakukan pengujian keamanan yang lebih komprehensif terhadap berbagai kemungkinan serangan serta melakukan pengujian performa dengan variasi jumlah permintaan sebelum sistem diterapkan pada lingkungan produksi.

Keenam, mekanisme OTP pada sistem saat ini menggunakan *session* sebagai media penyimpanan kode verifikasi. Pendekatan tersebut sesuai untuk menyimpan data OTP yang bersifat sementara, tetapi memiliki keterbatasan karena keberadaan kode OTP bergantung pada ketersediaan data *session*. Apabila data *session* yang menyimpan OTP hilang atau tidak tersedia, misalnya akibat kondisi tertentu pada pengelolaan *session* setelah *server restart*, pengguna yang sedang berada pada tahap verifikasi tidak dapat melanjutkan proses menggunakan OTP sebelumnya. Dalam kondisi tersebut, pengguna perlu melakukan proses autentikasi kembali atau meminta pengiriman OTP baru. Oleh karena itu, pengembangan selanjutnya disarankan untuk mempertimbangkan penggunaan media penyimpanan OTP yang lebih persisten atau terpusat, seperti basis data, dengan menerapkan waktu kedaluwarsa dan penghapusan OTP setelah berhasil digunakan. Hal tersebut diharapkan dapat meningkatkan keandalan proses verifikasi OTP pada lingkungan produksi.

DAFTAR PUSTAKA

Arianto, O. D., & Susetyo, Y. A. (2022). PENERAPAN RESTFUL WEB SERVICE DENGAN FRAMEWORK LARAVEL UNTUK PEMBANGUNAN SISTEM INFORMASI MANAJEMEN SUMBER DAYA MANUSIA. *JIPI (Jurnal Ilmiah Penelitian Dan Pembelajaran Informatika)*, *7*(2), 522–532. https://doi.org/10.29100/jipi.v7i2.2870

Herdiyatmoko, H. F. (2022). DESAIN SISTEM BACKEND BERBASIS REST API MENGGUNAKAN FRAMEWORK LARAVEL 7. *SKANIKA*, *5*(2), 136–144. https://doi.org/10.36080/skanika.v5i2.2947

Hermanto, A., Utama, A. N., Muflihah, Y., Januantoro, A., & Kusnanto, G. (2024). Pemanfaatan Metode Sinkron-Asinkron Komunikasi Data Dengan Flutter Sebagai Solusi Layanan Aplikasi Untuk Daerah Dengan Keterbatasan Sinyal Internet. *Jurnal Sistem Dan Teknologi Informasi (JustIN)*, *12*(3), 405. https://doi.org/10.26418/justin.v12i3.75164

Ndaru, A. D., Aljawari, M., & Setiawan, M. Y. (2025). Implementasi REST API Pengelolaan Data Penduduk Multi-Desa Berbasis Laravel dengan Autentikasi Sanctum. *Merkurius : Jurnal Riset Sistem Informasi Dan Teknik Informatika*, *3*(4), 325–335. https://doi.org/10.61132/merkurius.v3i4.997

Nurandani, D., Sitorus, S. P., Manoppo, A. R., Adrian, D., & Ayuda, M. I. (2025). Penggunaan OTP (One Time Password) Sebagai Lapisan Keamanan Tambahan dalam Aplikasi AY Pulsa. *Jurnal Minfo Polgan*, *14*(2), 3304–3314. https://doi.org/10.33395/jmp.v14i2.15724

Rachman, R., Kharisma, & Tolle, H. (2025). *ANALISIS PERBANDINGAN KINERJA DATABASE ROOM DAN SQLITE DALAM PENANGANAN DATA MULTIMEDIA PADA APLIKASI PENCATATAN* (Vol. 9, Number 10). http://j-ptiik.ub.ac.id

Rahmawati, L., & Sumarsono, S. (2024). Desain Pengembangan Website dengan Arsitektur Model View Controller pada Framework Laravel. *Jurnal Teknologi Dan Sistem Informasi Bisnis*, *6*(4), 785–790. https://doi.org/10.47233/jteksis.v6i4.1497

Ramadhan Purba, G., Rizal, & Afrillia, Y. (2025). KEAMANAN ENDPOINT API MENGGUNAKAN OAUTH2 PADA UNIT LAYANAN TERPADU UNIVERSITAS MALIKUSSALEH. *Rabit : Jurnal Teknologi Dan Sistem Informasi Univrab*, *10*(2), 1424–1434. https://doi.org/10.36341/rabit.v10i2.6543

Rizal, R., & Rahmatulloh, A. (2019). RESTFUL WEB SERVICE UNTUK INTEGRASI SISTEM AKADEMIK DAN PERPUSTAKAAN UNIVERSITAS PERJUANGAN. *JURNAL ILMIAH INFORMATIKA*, *7*(01), 54. https://doi.org/10.33884/jif.v7i01.1004

Zahra, S. N., Khaira, U., & Arsa, D. (2024). Metode Agile Scrum pada Rancang Bangun Sistem Informasi Manajemen Pelatihan Pegawai Perusahaan. *Journal of Information System,Graphics, Hospitality and Technology*, *6*(2), 79–91. https://doi.org/10.37823/insight.v6i2.403

LAMPIRAN



gamatechno.com

No: 46115/PCC/STF/III/2026

SERTIFIKAT

PT Gamatechno Indonesia memberikan sertifikat kepada:

CERTIFIED

Riyaldo Dwi Nopianto Suryo S

Telah melaksanakan Magang dari tanggal 22 September 2025
sampai dengan 21 Maret 2026 dengan baik.

Yogyakarta, 25 Maret 2026
Manager Business Dev. & Support

Yohanes Andri Wardhana