BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Pelaksanaan kerja magang pada AirNav Indonesia sebagai IT Internship pada Divisi Teknologi Informasi di bawah Direktur Teknik. Selama pelaksanaan magang, proses koordinasi dilakukan melalui grup WhatsApp dan dipantau secara langsung oleh pihak supervisor. Setiap tugas diberikan melalui media komunikasi tersebut atau disampaikan secara langsung pada saat pertemuan.

3.2 Tugas yang Dilakukan

Selama masa magang pada AirNav Indonesia, pengembangan aplikasi HAADES (Highly Accurate Aircraft Data Enhancement System) berfokus pada pengembangan backend sistem dan implementasi fitur-fitur baru. Berikut merupakan tugas yang dilakukan:

1. Mengembangkan backend API untuk sistem validasi data penerbangan overflying
2. Melakukan integrasi database dengan berbagai sumber data (FPS, AFTN, AIDC, ATS System)
3. Mengimplementasikan business logic untuk proses periodesasi dan pelaporan data
4. Melakukan optimasi query database dan performa sistem
5. Melakukan testing dan debugging terhadap fitur yang telah dibuat
6. Menyusun dokumentasi teknis API dan database schema

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang berlangsung pada periode 01 September 2025 hingga 31 Desember 2025, sebagaimana dirangkum dalam Tabel 3.1. Selama periode tersebut, kegiatan magang dilaksanakan sesuai dengan rencana dan pembagian tugas yang telah ditetapkan.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Dilakukan orientasi awal pengembangan sistem dengan memahami kebutuhan operasional aplikasi serta alur kerja teknis yang akan diterapkan. Diskusi teknis bersama tim dilakukan untuk menyamakan pemahaman terhadap sistem yang akan dikembangkan.
2	Fokus pada pendalaman TypeScript serta pemahaman konsep API dan alur komunikasi data antara backend dan frontend sebagai dasar pengembangan sistem berbasis web.
3, 4	Mempelajari dokumentasi sistem yang telah berjalan serta melakukan setup project frontend menggunakan Nuxt, Tailwind, dan TypeScript. Selain itu, sistem autentikasi menggunakan AirNav Auth dipelajari dan diimplementasikan hingga proses login dan validasi token berjalan dengan baik.
5	Mulai beralih ke pengembangan backend dengan mempelajari struktur project backend, melakukan riset teknologi backend seperti Prisma, serta melakukan setup awal environment backend dan koneksi database.
6	Pengembangan backend difokuskan pada refactor struktur project agar lebih modular. Konfigurasi schema Prisma, sinkronisasi database, serta implementasi dan pengujian fitur CRUD dasar dilakukan.
7, 8	Pengembangan fitur lanjutan backend dilakukan, meliputi search, pagination, validasi data, pengelolaan file import, serta pengembangan modul backend dan dashboard awal. Integrasi awal backend dan frontend juga dilakukan dengan penyesuaian format respons API.
9	Dilakukan penyempurnaan dashboard analytics dan implementasi penuh Flight Plan System (FPS). Refactor struktur API, perbaikan bug, serta pengujian sistem secara menyeluruh menjadi fokus utama.
	Lanjut di halaman berikutnya

Tabel 3.1 – lanjutan dari halaman sebelumnya

Minggu Ke -	Pekerjaan yang dilakukan
10	Peninjauan ulang modul backend dilakukan untuk persiapan integrasi dengan database final. Penyusunan skema database sementara, simulasi query, serta koordinasi teknis dengan mentor dilaksanakan.
11	Pengembangan dan penyempurnaan fitur OVF dilakukan, meliputi implementasi CRUD, pagination, refactor model dan service, serta pengujian agar sistem berjalan stabil.
12	Pengembangan fitur OVERFLYING, flight management, dan log-system dilakukan, termasuk pengolahan file CSV, validasi data, serta penyempurnaan logika backend.
13	Peninjauan dan penyesuaian seluruh endpoint backend dilakukan untuk kesiapan integrasi database final. Simulasi alur data backend–frontend dan evaluasi sistem bersama mentor juga dilakukan.
14	Mempelajari struktur dan arsitektur project aplikasi perkantoran, melakukan setup environment pengembangan, serta eksplorasi kode backend dan frontend untuk memahami alur sistem.
15	Pengembangan dan pengujian fitur CRUD pada sistem aplikasi perkantoran dilakukan, termasuk penyesuaian endpoint backend dan validasi data agar sistem berjalan stabil.
16	Dilakukan pengujian lanjutan, penyempurnaan fitur, serta integrasi antar modul untuk memastikan konsistensi data dan kestabilan sistem secara keseluruhan.
17	Pengembangan modul tambahan dan pengujian integrasi akhir dilakukan untuk memastikan seluruh sistem siap digunakan dan terintegrasi dengan baik.

3.3.1 Sistem Login Authentication AirNav

A Requirement (Kebutuhan Sistem)

Sistem autentikasi merupakan komponen fundamental dalam pengembangan HAADES yang memerlukan pengelolaan akses pengguna

secara aman dan terpusat. Kebutuhan sistem autentikasi HAADES meliputi:

1. Kebutuhan Fungsional:
    - Sistem harus dapat melakukan autentikasi pengguna menggunakan layanan Single Sign-On (SSO) terpusat AirNav Auth
    - Sistem harus dapat mengelola sesi pengguna dengan token berbasis waktu (time-based token)
    - Sistem harus dapat memperbarui token yang kedaluwarsa tanpa perlu login ulang
    - Sistem harus dapat memverifikasi validitas token pada setiap permintaan
    - Sistem harus dapat mengambil data profil pengguna yang terautentikasi
    - Sistem harus dapat melakukan proses logout dan menghapus sesi pengguna
2. Kebutuhan Non-Fungsional:
    - Keamanan: Token harus diencode menggunakan Base64 dan memiliki masa berlaku terbatas
    - Integrasi: Sistem harus terintegrasi dengan layanan AirNav Auth yang sudah ada
    - Konsistensi: Mekanisme autentikasi harus konsisten di seluruh modul aplikasi

B Perancangan Sistem Autentikasi

Perancangan sistem autentikasi HAADES menggunakan AirNav Auth sebagai layanan SSO terpusat yang dikembangkan oleh PT Angkasa Pura Aviasi. Alur perancangan sistem meliputi beberapa tahapan sebagai berikut:

1. Registrasi Aplikasi Tahap awal dalam perancangan adalah mendaftarkan redirect URI aplikasi ke dalam sistem AirNav Auth. Redirect URI yang didaftarkan dapat berupa alamat pengembangan lokal maupun alamat production. URI ini akan diencode menggunakan metode Base64 untuk keperluan keamanan. Sistem akan melakukan validasi terhadap redirect URI yang dikirimkan, dan jika URI terdaftar dalam basis data sistem, halaman login akan ditampilkan untuk proses autentikasi.

2. Mekanisme Token-Based Authentication Sistem dirancang menggunakan mekanisme token-based authentication yang menghasilkan tiga jenis token:

   - Access Token: Token dengan masa aktif 24 jam untuk mengakses sistem
   - Refresh Token: Token untuk memperbarui access token yang kedaluwarsa
   - Expire Token: Informasi waktu kedaluwarsa token

   Pendekatan ini memastikan stateless communication dan keamanan dengan expiration time otomatis untuk mencegah session hijacking.

3. Alur Autentikasi Alur autentikasi dirancang dengan flowchart yang menggambarkan proses dari login hingga logout (lihat Gambar 3.1). Proses dimulai dari pengguna mengakses aplikasi, sistem mengarahkan ke halaman login AirNav Auth, pengguna memasukkan kredensial, sistem memvalidasi dan menghasilkan token, token dikirim ke redirect URI, hingga pengguna dapat mengakses aplikasi.

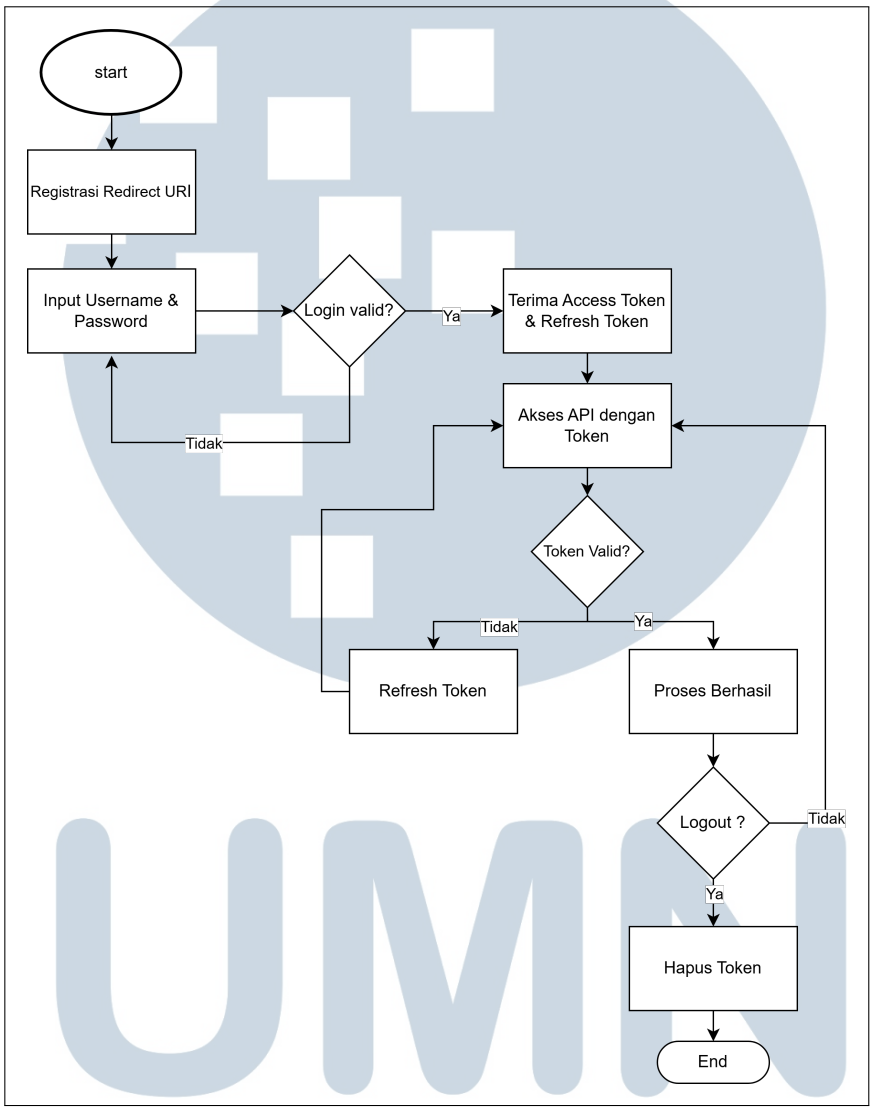
C Implementasi

Implementasi sistem autentikasi dilakukan dengan mengintegrasikan endpoint API dari AirNav Auth. Tahap awal dimulai dengan registrasi redirect URI aplikasi yang diencode menggunakan Base64 untuk keamanan. Proses login diimplementasikan melalui API endpoint dengan method POST yang menerima parameter username, password, dan redirect URI, kemudian menghasilkan tiga jenis token: Access Token (masa aktif 24 jam), Refresh Token, dan Expire Token yang dikirim ke redirect URI untuk disimpan dan digunakan pada permintaan berikutnya.

Sistem menyediakan mekanisme refresh token untuk memperpanjang masa aktif sesi tanpa login ulang dengan mengirimkan refresh token ke endpoint khusus menggunakan Authorization Bearer Token. Verifikasi token diimplementasikan melalui endpoint yang menerima Bearer Token dan mengembalikan status validitas token untuk memastikan keamanan akses. Pengambilan data profil pengguna dilakukan dengan mengirimkan Bearer Token ke endpoint profil yang mengembalikan data dalam format JSON, mencakup informasi seperti nama, email, dan role pengguna.

Proses logout diimplementasikan dengan mengakses endpoint logout yang menyertakan redirect URI terenkripsi Base64. Sistem akan menghapus sesi

pengguna, membatalkan validitas token, dan mengarahkan pengguna kembali ke halaman login, memastikan keamanan dengan mencegah penggunaan token setelah sesi berakhir.



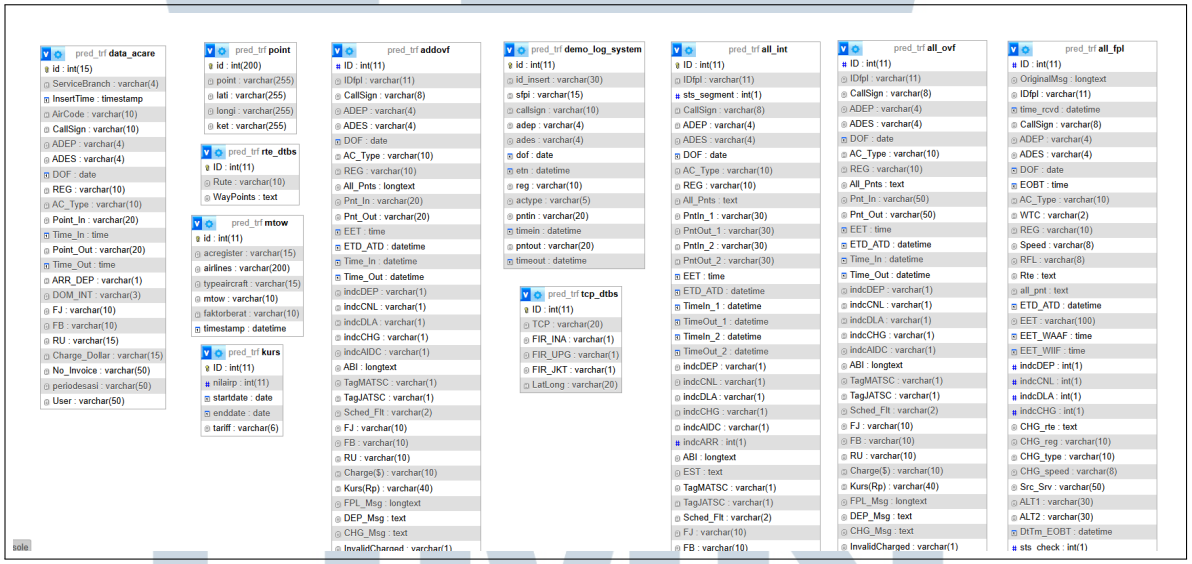
Gambar 3.1. *Flowchart* sistem login

3.3.2 Database

Perancangan database untuk aplikasi HAADES menggunakan MySQL sebagai Relational Database Management System (RDBMS) dengan Prisma ORM sebagai data access layer. Pemilihan relational database sangat tepat untuk sistem ini karena data penerbangan memiliki struktur yang terorganisir dengan baik dan banyak relasi antar entitas (penerbangan, pesawat, waypoint, rute, dll). Database

dirancang untuk menampung 11 tabel utama yang mendukung 13 modul fungsional sistem.

Database design mengikuti prinsip normalisasi hingga *Third Normal Form* (3NF) untuk mengurangi redundansi data dan menjaga integritas. Namun, untuk tabel-tabel transaksional yang sangat besar seperti all_fpl, all_ovf, dan all_int, dilakukan sedikit denormalisasi dengan menyimpan beberapa computed fields atau agregat untuk optimasi performa query. Strategi indexing diterapkan secara selektif pada kolom-kolom yang sering digunakan untuk *searching* dan *filtering*, seperti CallSign, ADEP, ADES, DOF (Date of Flight), dan timestamp fields. Penerapan indexing ini menjadi krusial mengingat tabel-tabel transaksional tersebut dapat memuat ratusan ribu hingga jutaan *records* untuk data *historis*.



Gambar 3.2. Tabel Database Haades

1. Tabel all_fpl (Flight Plan Data)

   Tabel all_fpl menyimpan data lengkap flight plan yang diajukan ke dalam sistem dan berperan sebagai tabel inti pada modul Flight Plan System (FPS). Tabel ini merupakan tabel paling komprehensif dalam HAADES dengan lebih dari 55 field yang mencakup seluruh aspek flight plan, termasuk penyimpanan pesan AFTN asli pada field OriginalMsg untuk keperluan audit trail. Setiap flight plan memiliki pengenal unik IDfpl yang digunakan sebagai referensi dan pelacakan di seluruh sistem.

   Field indikator seperti indcDEP, indcCNL, indcDLA, indcCHG, dan indcARR berfungsi sebagai penanda status untuk melacak siklus hidup

penerbangan berdasarkan pesan AFTN yang diterima, sehingga status terkini penerbangan dapat diketahui tanpa melakukan pemrosesan ulang pesan. Field dengan awalan CHG_ digunakan untuk menyimpan data perubahan dari pesan Change, memungkinkan penyimpanan data awal dan data hasil perubahan secara bersamaan. Selain itu, field pnt_in, pnt_out, timein, dan timeout dengan penanda segmen digunakan untuk mendukung penerbangan two-segment yang melintasi FIR Indonesia lebih dari satu kali.

Mengingat volume data yang sangat besar, tabel ini memerlukan strategi pengindeksan yang optimal. Indeks pada CallSign, ADEP, ADES, dan DOF digunakan untuk mendukung pencarian dan penyaringan data, sementara indeks unik pada IDfpl menjamin keunikan identitas flight plan. Kombinasi indeks pada CallSign dan DOF memberikan peningkatan performa pada pola kueri pencarian penerbangan berdasarkan tanggal tertentu.

Tabel 3.2. Struktur Field Tabel Flight Plan (FPL)

Field	**Type**	**Deskripsi**
ID	INT	Primary key, auto increment
OriginalMsg	LONGTEXT	Original AFTN message yang diterima
IDfpl	VARCHAR(11)	Unique flight plan identifier
time_rcvd	DATETIME	Waktu message diterima sistem
CallSign	VARCHAR(8)	Flight callsign (contoh: “GIA123”)
ADEP	VARCHAR(4)	Departure airport ICAO code (contoh: “WIII”)
ADES	VARCHAR(4)	Destination airport ICAO code (contoh: “WSSS”)
DOF	DATE	Date of flight
EOBT	TIME	Estimated off-block time
AC_Type	VARCHAR(10)	Aircraft type (contoh: “B738”, “A320”)
WTC	VARCHAR(2)	Wake turbulence category (L/M/H/J)
REG	VARCHAR(10)	Aircraft registration
Speed	VARCHAR(8)	Cruising speed
RFL	VARCHAR(8)	Requested flight level
Rte	TEXT	Complete route string
all_pnt	TEXT	Seluruh waypoint dalam rute
		Lanjut di halaman berikutnya

Tabel 3.2 – lanjutan dari halaman sebelumnya

Field	Type	Deskripsi
ETD_ATD	DATETIME	Estimated/Actual time of departure
EET	VARCHAR(100)	Estimated elapsed time
EET_WAAF	TIME	Estimated elapsed time pada FIR WAAF
EET_WIIF	TIME	Estimated elapsed time pada FIR WIIF
indcDEP	INT	Indikator pesan DEP diterima (0/1)
indcCNL	INT	Indikator pesan CNL diterima (0/1)
indcDLA	INT	Indikator pesan DLA diterima (0/1)
indcCHG	INT	Indikator pesan CHG diterima (0/1)
indcARR	INT	Indikator pesan ARR diterima (0/1)
CHG_rte	TEXT	Perubahan rute dari pesan CHG

Mengingat struktur tabel all_fpl yang sangat kompleks dengan lebih dari 55 atribut, Tabel di atas disederhanakan untuk hanya menampilkan field-field esensial yang memiliki relevansi langsung dengan logika bisnis backend yang dibahas.

2. Tabel all_ovf (Overflying Flight Data)

   Tabel all_ovf menyimpan data penerbangan overflying, yaitu penerbangan yang melintasi FIR Indonesia tanpa melakukan pendaratan, dan digunakan oleh modul Flight Plan System (FPS) serta perhitungan billing. Sebagai tabel inti dalam HAADES, tabel ini terdiri dari 67 field yang mencakup data penerbangan, titik masuk dan keluar FIR (Pnt_In dan Pnt_Out), informasi penagihan, serta berbagai penanda status untuk pengelolaan alur kerja. Field Pnt_In dan Pnt_Out berperan penting sebagai dasar perhitungan jarak dan biaya overflight.

   Modul FPS sangat bergantung pada field yang diawali dengan fps_. Field fps_sts menunjukkan status aktivasi FPS, sementara field Validated berfungsi sebagai mekanisme validasi untuk memastikan hanya data yang telah diverifikasi yang dapat diproses lebih lanjut ke tahap penagihan. Field fps_updater dan fps_time_updated digunakan untuk mencatat riwayat pembaruan data sebagai bagian dari audit trail.

   Komponen billing mencakup field hasil perhitungan seperti FJ, FB, RU, dan Charge yang disimpan secara permanen sebagai bentuk denormalisasi guna

menjaga konsistensi data historis dan meningkatkan performa kueri. Nilai kurs pada saat perhitungan disimpan pada field Kurs(Rp) agar perubahan nilai tukar di masa mendatang tidak memengaruhi data penagihan yang telah final.

Selain itu, tabel ini menerapkan mekanisme soft delete melalui field StatusDelete untuk mendukung kebutuhan audit trail, serta field stamp_period dan berbagai status flag lainnya untuk mendukung pengelolaan alur kerja pelaporan dan penagihan yang kompleks.

Tabel 3.3. Struktur Field Tabel OVF

Field	**Type**	**Deskripsi**
ID	INT	Primary key
IDfpl	VARCHAR(11)	Flight plan identifier, default “0”
CallSign	VARCHAR(8)	Flight callsign
ADEP	VARCHAR(4)	Departure airport
ADES	VARCHAR(4)	Destination airport
DOF	DATE	Date of flight
AC_Type	VARCHAR(10)	Aircraft type
REG	VARCHAR(10)	Aircraft registration
All_Pnts	TEXT	Seluruh point dalam rute
Pnt_In	VARCHAR(50)	Entry point ke FIR Indonesia
Pnt_Out	VARCHAR(50)	Exit point dari FIR Indonesia
EET	TIME	Estimated elapsed time
ETD_ATD	DATETIME	Estimated/Actual time of departure
Time_In	DATETIME	Waktu masuk FIR Indonesia
Time_Out	DATETIME	Waktu keluar FIR Indonesia
indcDEP	VARCHAR(1)	DEP indicator (0/1)
indcCNL	VARCHAR(1)	CNL indicator
indcDLA	VARCHAR(1)	DLA indicator
indcCHG	VARCHAR(1)	CHG indicator

Serupa dengan tabel all_fpl, tabel all_ovf terdiri dari 67 field yang mencakup detail data penerbangan lintas wilayah. Untuk efisiensi penyajian dalam laporan ini, tabel di atas ini merepresentasikan kolom-kolom yang relevan dengan implementasi modul OVF, tanpa mengurangi substansi alur data yang digunakan.

3. Tabel all_int (International 2-Segment Flight Data)

   Tabel ini menyimpan data penerbangan internasional yang memiliki dua segmen terpisah dalam FIR Indonesia, yaitu penerbangan yang masuk dan keluar wilayah FIR lebih dari satu kali, seperti rute Singapura–Bali dan Bali–Jakarta. Selain field khusus untuk pengelolaan dua segmen, tabel ini juga memuat seluruh field yang terdapat pada tabel all_ovf, termasuk informasi penerbangan, data pesawat, komponen billing, serta berbagai penanda status.

   Tingkat kompleksitas tabel ini lebih tinggi dibandingkan tabel lainnya karena harus mendukung pelacakan dua segmen masuk dan keluar FIR secara terpisah. Proses validasi memastikan bahwa waktu keluar segmen pertama terjadi sebelum waktu masuk segmen kedua (TimeOut_1 ¡ TimeIn_2). Selain itu, perhitungan billing dilakukan untuk masing-masing segmen, serta didukung oleh penanganan pesan kedatangan (arrival message) karena tujuan akhir penerbangan berada di wilayah Indonesia.

Tabel 3.4. Struktur Field Tabel 2-Segment Flight Data

Field	**Type**	**Deskripsi**
sts_segment	INT	Segment status (0/1/2)
PntIn_1	VARCHAR(30)	Entry point segment 1
PntOut_1	VARCHAR(30)	Exit point segment 1
PntIn_2	VARCHAR(30)	Entry point segment 2
PntOut_2	VARCHAR(30)	Exit point segment 2
TimeIn_1	DATETIME	Time masuk segment 1
TimeOut_1	DATETIME	Time keluar segment 1
TimeIn_2	DATETIME	Time masuk segment 2
TimeOut_2	DATETIME	Time keluar segment 2
indcARR	INT	Arrival indicator
ARR_Msg	TEXT	AFTN ARR message

4. Tabel Mtow

   Tabel mtow berfungsi menyimpan informasi berat maksimum lepas landas untuk berbagai jenis pesawat, yang merupakan salah satu faktor kritis dalam perhitungan billing overflight. Field faktorberat digunakan langsung dalam formula perhitungan tarif, sementara acregister dan typeaircraft

memungkinkan sistem untuk mencari data MTOW berdasarkan registrasi atau tipe pesawat yang tercantum dalam flight plan.

Tabel 3.5. Struktur Field Tabel Mtow

Field	**Type**	**Deskripsi**
rs_id	INT	Primary key
acregister	VARCHAR(15)	Aircraft registration
airlines	VARCHAR(200)	Airlines name
typeaircraft	VARCHAR(15)	Aircraft type
mtow	VARCHAR(10)	MTOW value (in tons)
faktorberat	VARCHAR(10)	Weight factor untuk calculation
timestamp	DATETIME	Last update timestamp

5. Tabel Point

   Tabel point berfungsi yang menyimpan seluruh waypoint navigasi dalam wilayah FIR Indonesia beserta koordinat geografisnya. Data waypoint ini sangat penting untuk validasi titik masuk dan keluar (entry/exit points) penerbangan overflying, serta digunakan dalam perhitungan jarak tempuh untuk penentuan tarif.

Tabel 3.6. Struktur Field Tabel Point

Field	**Type**	**Deskripsi**
rs_id	INT	Primary key
point	VARCHAR(255)	Point name/identifier
lati	VARCHAR(255)	Latitude
longi	VARCHAR(255)	Longitude
ket	VARCHAR(255)	Keterangan/remarks

6. Tabel rte_dtbs (Route)

   Tabel rte_dtbs menyimpan master data rute penerbangan standar yang berisi sekuens waypoint untuk berbagai rute yang umum digunakan. Tabel ini memfasilitasi validasi rute penerbangan dan memberikan kemudahan auto-complete saat operator memasukkan data flight plan, meningkatkan efisiensi data entry dan mengurangi kesalahan input. Digunakan untuk validasi rute dan auto-complete saat input flight plan.

Tabel 3.7. Struktur Field Tabel Route

Field	Type	Deskripsi
id	INT	Primary key
rute	VARCHAR(10)	Route name/identifier
wayPoints	TEXT	Sequence of waypoints

7. Tabel tcp_dtbs (TCP Boundary)

   Tabel tcp_dtbs menyimpan informasi Traffic Control Points yang merupakan titik-titik batas antar FIR (Flight Information Region), dilengkapi dengan flags untuk mengidentifikasi FIR mana yang mencakup setiap TCP. Data ini digunakan untuk menentukan zona billing dan memvalidasi bahwa titik masuk/keluar penerbangan berada pada batas FIR yang benar. Untuk validasi entry/exit points dan billing zone determination.

Tabel 3.8. Struktur Field Tabel Data TCP

Field	Type	Deskripsi
id	INT	Primary key
tcp	VARCHAR(20)	TCP name
fir_ina	VARCHAR(1)	Flag: dalam FIR Indonesia (Y/N)
fir_upg	VARCHAR(1)	Flag: dalam FIR Ujung Pandang (Y/N)
fir_jkt	VARCHAR(1)	Flag: dalam FIR Jakarta (Y/N)
latLong	VARCHAR(20)	Koordinat lat/long

8. Tabel kurs

   Tabel kurs menyimpan nilai tukar mata uang Rupiah terhadap dolar Amerika yang berlaku untuk periode tertentu, dengan range tanggal mulai dan berakhirnya setiap nilai kurs. Sistem secara otomatis memilih nilai kurs yang sesuai berdasarkan tanggal penerbangan (DOF) untuk memastikan akurasi perhitungan billing dalam Rupiah, dan nilai kurs yang digunakan disimpan dalam tabel transaksional untuk menjaga konsistensi historical data meskipun nilai kurs berubah di masa mendatang.

Tabel 3.9. Struktur Field Tabel Kurs

Field	Type	Deskripsi
id	INT	Primary key
nilairp	INT	Nilai Rupiah per 1 USD
startdate	DATE	Tanggal mulai berlaku
enddate	DATE	Tanggal akhir berlaku
tariff	VARCHAR(6)	Kode tarif

9. Tabel data_acare (A-CARE)

   Tabel data_acare dirancang khusus untuk menampung data penerbangan dari sistem A-CARE (Arrival/Departure Charging) dengan dukungan fitur bulk upload melalui file Excel. Tabel ini mencakup informasi lengkap penerbangan arrival dan departure beserta data billing seperti factors (FJ, FB, RU) dan nilai tagihan, serta dilengkapi dengan informasi invoice dan periode sasi untuk keperluan administrasi dan pelaporan.

Tabel 3.10. Struktur Field Tabel A-Care

Field	Type	Deskripsi
rs_id	INT	Primary key
ServiceBranch	VARCHAR(4)	Cabang service
InsertTime	TIMESTAMP	Auto-updated timestamp
AirCode, CallSign	VARCHAR(10)	Flight identifiers
ADEP, ADES	VARCHAR(4)	Departure/destination
DOF	DATE	Date of flight
REG, AC_Type	VARCHAR(10)	Aircraft info
Point_In, Point_Out	VARCHAR(20)	Entry/exit points
Time_In, Time_Out	TIME	Entry/exit times
ARR_DEP	VARCHAR(1)	Arrival or Departure flag
DOM_INT	VARCHAR(3)	Domestic or International
FJ, FB, RU	VARCHAR	Billing factors
Charge_Dollar	VARCHAR(15)	Charge amount
No_Invoice	VARCHAR(50)	Invoice number
periodesasi	VARCHAR(50)	Periode sesi
User	VARCHAR(50)	User yang input

10. Tabel demo_log_system (Log System Data)

Tabel demo_log_system berfungsi sebagai repositori log sistem yang merekam aktivitas dan transaksi untuk keperluan monitoring, tracking, dan auditing. Dengan dukungan bulk upload melalui CSV dan strategi indexing yang agresif pada 5 field utama (callsign, adep, ades, dof, reg), tabel ini dioptimasi untuk pencarian dan filtering yang cepat meskipun berisi volume data log yang sangat besar.

Tabel 3.11. Struktur Field Tabel Log System Data

Field	**Type**	**Deskripsi**
rs_id	INT	Primary key
id_insert	VARCHAR(30)	Insert identifier
sfpi	INT	SFPI value
callsign	VARCHAR(10)	Flight callsign
adep, ades	VARCHAR(4)	Airports
dof	DATE	Date of flight
etn	DATETIME	ETN time
reg	VARCHAR(10)	Registration
actype	VARCHAR(5)	Aircraft type
pntin, pntout	VARCHAR(20)	Points
timein, timeout	DATETIME	Times

11. Tabel addovf (addovf Data)

    Tabel addovf berfungsi sebagai tabel pendukung yang menyimpan data tambahan untuk penerbangan overflying guna melengkapi informasi pada tabel all_ovf. Tabel ini digunakan untuk mencatat informasi operasional tertentu yang tidak dapat diakomodasi dalam struktur tabel utama.

Tabel 3.12. Struktur Field Tabel Tabel Addovf

Field	**Tipe Data**	**Deskripsi**
ID	INT	Identifier unik sebagai *logical primary key*
IDfpl	VARCHAR(11)	Identifier *flight plan*
CallSign	VARCHAR(8)	*Flight callsign*
ADEP	VARCHAR(4)	*Aerodrome of Departure*
ADES	VARCHAR(4)	*Aerodrome of Destination*
DOF	DATE	*Date of Flight*
AC_Type	VARCHAR(10)	*Aircraft Type*
REG	VARCHAR(10)	*Aircraft Registration*
Pnt_In, Pnt_Out	VARCHAR(20)	*Entry* dan *Exit Points*
Time_In, Time_Out	DATETIME	Waktu masuk dan keluar FIR
FJ, FB, RU	VARCHAR(10)	Faktor Jarak, Berat, dan *Route Unit*
Charge ($)	VARCHAR(10)	Total biaya dalam USD
fps_sts	VARCHAR(1)	Status FPS (0/1)
Validated	VARCHAR(1)	Status validasi data

Tabel ini memiliki relasi satu-ke-satu opsional dengan all_ovf melalui IDfpl serta kombinasi CallSign, ADEP, ADES, dan DOF. Data bersifat dinamis dan hanya dicatat apabila diperlukan, dengan mekanisme soft delete melalui field Status serta pengelolaan timestamp otomatis oleh basis data.

3.3.3 Perancangan sistem Backend

A Perancangan API (REST API)

Perancangan Application Programming Interface (API) pada sistem HAADES Backend menggunakan arsitektur Representational State Transfer (REST) sebagai standar komunikasi antara klien dan server. Penerapan REST memastikan API bersifat scalable, maintainable, dan mudah diintegrasikan oleh developer. REST API memanfaatkan protokol HTTP dengan metode standar CRUD (GET, POST, PUT, DELETE), serta menggunakan Uniform Resource Identifier (URI) yang konsisten dan deskriptif untuk merepresentasikan resource seperti Flight Plan, MTOW, dan Route.

Implementasi prinsip REST pada HAADES Backend mencakup penggunaan resource-based URL yang terstruktur, komunikasi stateless pada setiap request, serta format response yang distandardisasi secara konsisten di seluruh endpoint.

1. Standardisasi Response

   Standardisasi struktur response merupakan aspek krusial dalam memastikan konsistensi komunikasi API. Sistem HAADES Backend menerapkan tiga jenis response format standar yang disesuaikan dengan konteks operasi yang dilakukan. Success Response digunakan untuk operasi yang berhasil dieksekusi. Struktur response mencakup flag success bernilai true, field message yang berisi deskripsi hasil operasi, dan field data yang mengandung payload informasi yang diminta. Format ini memastikan klien dapat dengan mudah mengidentifikasi status keberhasilan operasi dan mengekstrak data yang diperlukan.

```
{
  "success": true,
  "message": "Operation successful",
  "data": { ... }
}
```

Kode 3.1: Contoh format Success Respons

   Error Response diterapkan ketika terjadi kegagalan dalam pemrosesan request. Response ini memiliki flag success bernilai false, field message yang menjelaskan nature dari error yang terjadi, dan field errors yang berisi array detail kesalahan. Field errors khususnya berguna untuk validation errors dimana multiple fields dapat mengalami kegagalan validasi secara simultan.

```
{
  "success": false,
  "message": "Error description",
  "errors": [ ... ]
}
```

Kode 3.2: Contoh format Error Response

   Pagination Response diimplementasikan untuk endpoint yang mengembalikan koleksi data dalam jumlah besar. Selain struktur success response standar, format ini menambahkan object pagination yang berisi metadata tentang total records, halaman saat ini, limit per halaman, dan total

halaman yang tersedia. Pagination memastikan performa optimal dengan membatasi volume data yang ditransfer dalam single request.

```
{
  "success": true,
  "data": [ ... ],
  "pagination": {
    "total": 100,
    "page": 1,
    "limit": 10,
    "totalPages": 10
  }
}
```

Kode 3.3: Contoh format Pagination Response

2. API Endpoints

- Dashboard

  Endpoint /api/dashboard menggunakan metode GET untuk mengambil data agregat dan statistik sistem tanpa memerlukan autentikasi. Response berupa objek tunggal yang berisi berbagai metrik numerik seperti totalFPS, totalOverflying, total2Segment, dan array monthlyStats untuk visualisasi tren. Endpoint ini tidak mendukung paginasi karena hanya mengembalikan satu set data ringkasan yang sudah diproses server-side.

Tabel 3.13. Endpoint modul Dashboard

Metode	**Endpoint**	**Deskripsi Singkat**
GET	/api/dashboard	Get dashboard statistics

- MTOW Management

  Endpoint MTOW menyediakan operasi CRUD lengkap dengan GET /api/mtow mendukung query parameters page dan size untuk paginasi. POST dan PUT endpoint memvalidasi input menggunakan schema Zod yang memastikan field acregister, airlines, typeaircraft, mtow, dan faktorberat sesuai format. Response sukses mengembalikan objek MTOW lengkap dengan ID auto-generated, sementara error validation mengembalikan array errors dengan detail field yang bermasalah.

Tabel 3.14. Endpoint modul MTOW

Metode	Endpoint	Deskripsi Singkat
GET	/api/mtow	Get all MTOW data
GET	/api/mtow/:id	Get MTOW by ID
POST	/api/mtow	Create new MTOW
PUT	/api/mtow/:id	Update MTOW
DELETE	/api/mtow/:id	Delete MTOW

- Point Management

  Endpoint Point mengikuti pola RESTful standar dengan GET /api/point untuk list, GET /api/point/:id untuk detail, POST untuk create dengan required fields point (nama waypoint), lati, longi, dan optional field ket untuk keterangan. PUT endpoint menerima partial update, memungkinkan perubahan hanya pada field tertentu tanpa mengirim seluruh objek. DELETE melakukan hard delete langsung dari database.

Tabel 3.15. Endpoint modul Point

Metode	Endpoint	Deskripsi Singkat
GET	/api/point	Get all points data
GET	/api/point:id	Get points by ID
POST	/api/point	Create new points
PUT	/api/point:id	Update points
DELETE	/api/point:id	Delete points

- Route Management

  Endpoint Route mengelola data dengan POST request yang menerima Rute (identifier maksimal 10 karakter) dan WayPoints (TEXT field berisi sekuens waypoint separated by space). GET endpoint mengembalikan array objek route dengan struktur sederhana berisi ID, Rute, dan WayPoints. Update via PUT mendukung modifikasi sekuens waypoint dalam field WayPoints tanpa perlu recreate route.

Tabel 3.16. Endpoint modul Route

Metode	Endpoint	Deskripsi Singkat
GET	/api/route	Get all route data
GET	/api/route:id	Get route by ID
POST	/api/route	Create new route
PUT	/api/route:id	Update route
DELETE	/api/route:id	Delete route

- TCP Boundary Management

  Endpoint TCP Boundary memiliki struktur request dengan field TCP (nama TCP), FIR_INA, FIR_UPG, FIR_JKT (flags Y/N untuk identifikasi FIR), dan LatLong untuk koordinat. Response GET mengembalikan objek dengan semua field termasuk flags yang digunakan untuk filtering dan logic penentuan zona billing. Endpoint ini critical untuk validasi entry/exit points dalam FPS validation workflow.

Tabel 3.17. Endpoint modul TCP Boundary

Metode	Endpoint	Deskripsi Singkat
GET	/api/tcp-boundary	Get all TCP boundaries
GET	/api/tcp-boundary:id	Get TCP by ID
POST	/api/tcp-boundary	Create new TCP
PUT	/api/tcp-boundary:id	Update TCP
DELETE	/api/tcp-boundary:id	Delete TCP

- Kurs Management

  Endpoint Kurs menerima POST request dengan field nilairp (integer), startdate, enddate (format YYYY-MM-DD), dan tariff (kode tarif max 6 char). GET endpoint mendukung filtering berdasarkan date range untuk mendapatkan kurs yang berlaku pada periode tertentu. System logic memilih kurs berdasarkan date range checking startdate DOF enddate saat billing calculation.

Tabel 3.18. Endpoint modul Kurs

Metode	Endpoint	Deskripsi Singkat
GET	/api/kurs	Get all kurs
GET	/api/kurs:id	Get kurs by ID
POST	/api/kurs	Create new kurs
PUT	/api/kurs:id	Update kurs
DELETE	/api/kurs:id	Delete kurs

- A-CARE Management

  Endpoint A-CARE mencakup CRUD standar plus dua endpoint khusus: POST /api/acare/upload menerima multipart/form-data dengan field file (.xlsx/.xls) yang diproses menggunakan library XLSX, memvalidasi setiap row, dan mengembalikan response dengan inserted count dan optional errors array untuk rows yang gagal. POST /api/acare/delete-period menerima JSON body ”periodesasi”: ”YYYY-MM” dan melakukan bulk delete, returning deletedCount dalam response.

Tabel 3.19. Endpoint modul A-Care

Metode	Endpoint	Deskripsi Singkat
GET	/api/acare	Get all A-CARE data
GET	/api/acare:id	Get A-CARE by ID
POST	/api/acare	Create new A-CARE
PUT	/api/acare	Update A-CARE
DELETE	/api/acare:id	Delete A-CARE

- FPS (Flight Plan System)

  Endpoint FPS merupakan API paling kompleks dalam sistem dengan 10 endpoints yang mencakup operasi CRUD standar plus 5 endpoints khusus untuk workflow validasi. GET /api/fps mendukung pagination (page, size) dan search filtering (search query param untuk CallSign, ADEP, ADES). Endpoint khusus /api/fps/:ID/validation mengembalikan data pre-filled untuk form validasi dengan beberapa field flagged sebagai read-only. PUT /api/fps/:ID/validate menerapkan business rules validation termasuk pengecekan waypoint existence di

database dan validasi temporal (Time_Out ¿ Time_In), mengembalikan error 400 dengan detail validation failures jika ada. Endpoint /api/fps/:ID/status mengontrol aktivasi FPS dengan toggle fps_sts antara ”0” (inactive) dan ”1” (active/green), dengan business rule bahwa hanya FPS tervalidasi (Validated = ’1’) yang dapat diaktifkan. GET /api/fps/:ID/aftn mengambil AFTN messages (FPL, DEP, CHG, ARR) dalam format text untuk display, sementara GET /api/fps/aftn/data menyediakan search/filter berdasarkan status green/red (determined by DEP_Msg presence).

Tabel 3.20. Endpoint modul FPS

Metode	**Endpoint**	**Deskripsi Singkat**
GET	/api/fps	List FPS with pagination and search
GET	/api/fps:id	Get FPS by ID
POST	/api/fps	Create new FPS
PUT	/api/fps/:ID	Update FPS
DELETE	/api/fps/:ID	Delete FPS
GET	/api/fps/:ID/validation	Get data for validation form
PUT	/api/fps/:ID/validate	Validate FPS data
PUT	/api/fps/:ID/status	Update FPS status (green/default)
GET	/api/fps/:ID/aftn	Get AFTN messages
GET	/api/fps/aftn/data	Get AFTN data with status

- Overflying Management

  Endpoint Overflying menyediakan operasi CRUD standar dengan GET /api/overflying mendukung query parameters untuk pagination (page, size) dan search filtering. Response structure mengikuti format standar dengan field success, data array, dan pagination object. POST dan PUT endpoint menerima objek dengan fields standar penerbangan termasuk CallSign, ADEP, ADES, DOF, dan AC_Type.

Tabel 3.21. Endpoint modul Overflying

Metode	Endpoint	Deskripsi Singkat
GET	/api/overflying	Get all overflying data
GET	/api/overflying:id	Get overflying by ID
POST	/api/overflying	Create new overflying
PUT	/api/overflying	Update overflying
DELETE	/api/overflying:id	Delete overflying

- OVF RTB Management

  Endpoint OVF RTB mengikuti struktur API yang identik dengan Overflying module namun dengan URL path /api/ovf-rtb. Request dan response formats sama, perbedaan hanya pada filtering logic server-side yang fokus pada penerbangan dengan flag rtb_sts tertentu. Endpoint ini memudahkan frontend untuk mendapatkan data RTB tanpa perlu filtering manual.

Tabel 3.22. Endpoint modul OVF RTB

Metode	Endpoint	Deskripsi Singkat
GET	/api/ovf-rtb	Get all OVF RTB data
GET	/api/ovf-rtb:id	Get OVF RTB by ID
POST	/api/ovf-rtb	Create new OVF RTB
PUT	/api/ovf-rtb	Update OVF RTB
DELETE	/api/ovf-rtb:id	Delete OVF RTB

- OVF Local Management

  Endpoint OVF Local dengan path /api/ovf-local menggunakan request/response contract yang konsisten dengan OVF variants lainnya. Query parameters, request body structure, dan response format identik untuk memudahkan code reusability di frontend. Server-side filtering dibedakan berdasarkan kategori local flight yang stored dalam database flags.

Tabel 3.23. Endpoint modul OVF Local

Metode	Endpoint	Deskripsi Singkat
GET	/api/ovf-local	Get OVF Local data
GET	/api/ovf-local:id	OVF Local by ID
POST	/api/ovf-local	Create new OVF Local
PUT	/api/ovf-local	Update OVF Local
DELETE	/api/ovf-local:id	Delete OVF Local

- Flight 2-Segment Management

  Endpoint Flight 2-Segment memiliki request body yang lebih kompleks dengan fields berpasangan untuk dua segmen: PntIn_1, PntOut_1, TimeIn_1, TimeOut_1 untuk segmen pertama dan PntIn_2, PntOut_2, TimeIn_2, TimeOut_2 untuk segmen kedua. Validation logic memastikan TimeOut_1 TimeIn_2 (segmen 1 selesai sebelum segmen 2 dimulai). Response GET mengembalikan objek dengan semua paired fields.

Tabel 3.24. Endpoint modul Flight 2-Segment Management

Metode	Endpoint	Deskripsi Singkat
GET	/api/flight-2segment	List 2-segment flights
GET	/api/flight-2segment:id	GET by ID
POST	/api/flight-2segment	Create
PUT	/api/flight-2segment	Update
DELETE	/api/flight-2segment:id	Delete

- Log System

  Endpoint Log System terdiri dari GET /api/log-system dengan mandatory pagination (page, size query params) karena volume data log yang sangat besar, dan POST /api/log-system/upload yang menerima CSV file via multipart/form-data. Upload endpoint memproses CSV row-by-row, melakukan bulk insert ke database, dan mengembalikan statistics tentang jumlah records yang berhasil di-insert.

Tabel 3.25. Endpoint modul Log System

Metode	Endpoint	Deskripsi Singkat
GET	/api/log-system	Get log system list
POST	/api/log-system/upload	Upload CSV log file

B Perancangan Validasi dan Business Logic

1. Validasi Input

   Validasi input merupakan aspek penting dalam pengembangan HAADES Backend untuk menjaga integritas data, keamanan, dan keandalan sistem. Validasi dilakukan secara komprehensif menggunakan Zod sebagai pustaka utama yang mendukung runtime validation, type safety, serta penanganan kesalahan terstruktur. Zod dipilih karena kemampuannya memastikan kesesuaian data dengan skema yang telah ditetapkan serta menyediakan compile-time type safety melalui type inference pada TypeScript, sehingga konsistensi antara aturan validasi dan definisi tipe tetap terjaga [7]. Selain itu, Zod mendukung antarmuka yang intuitif, pesan kesalahan yang dapat dikustomisasi, serta komposisi dan penggunaan ulang skema validasi, termasuk penerapan metode partial() dan berbagai jenis validasi standar maupun kustom.

```
// 1. Definisikan Skema Validasi
SCHEMA FPS_Validation = OBJECT({
    CallSign: STRING().MIN(1).MAX(8),
    ADEP: STRING().MAX(4).OPTIONAL(),
    Time_In: STRING()
        .TRANSFORM((val) => val.REPLACE(".", ":"))
        .REFINE(IS_VALID_TIME_FORMAT)
})

// 2. Middleware Validasi Global
FUNCTION validate(schema) {
    RETURN (req, res, next) => {
        TRY {
            VALIDATED_DATA = schema.PARSE(req.body)
            req.body = VALIDATED_DATA // Menggunakan data hasil transformasi
            NEXT()
        } CATCH (error) {
            IF error IS ZodError {
                RETURN res.STATUS(400).JSON({
                    success: false,
                    errors: error.FORMAT()
                })
```

```
23                  }
24                  NEXT(error)
25              }
26          }
27      }
28
```

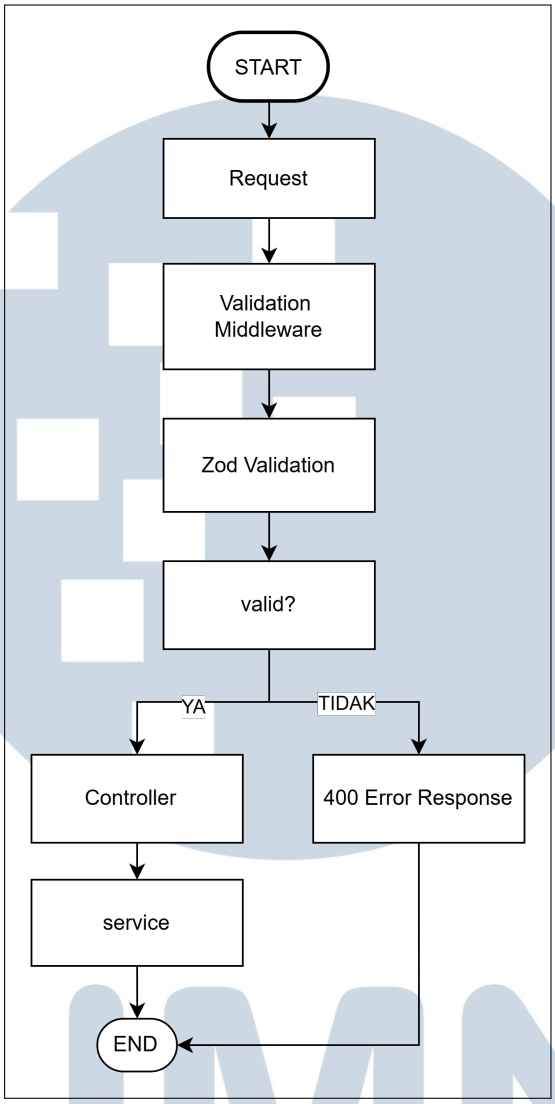
Kode 3.4: Skema Validasi dan Middleware Validasi Global pada Modul FPS

Validasi dimulai di middleware sebelum mencapai Controller. Zod melakukan transformasi otomatis (seperti mengubah format waktu dari ”13.40” menjadi ”13:40”) sehingga Service layer hanya menerima data yang sudah bersih. Jika terjadi kesalahan, middleware secara otomatis mengirimkan response 400 Bad Request berisi daftar error yang informatif bagi klien.

2. Arsitektur Validasi

Arsitektur validasi pada HAADES Backend dirancang dengan memisahkan logika validasi dari logika bisnis dan lapisan presentasi. Skema validasi ditempatkan dalam direktori validation/ dan disusun berdasarkan modul fungsional serta jenis use case, seperti create, update, get by ID, dan search. Pendekatan ini meningkatkan keteraturan struktur kode serta mendukung penggunaan ulang skema validasi pada berbagai controller.

Validasi diterapkan melalui middleware yang bersifat reusable dan menggunakan skema Zod untuk memeriksa data permintaan yang berasal dari request body, parameter, maupun query. Alur validasi dimulai pada tingkat router, di mana permintaan yang lolos validasi diteruskan ke lapisan controller dan service. Sebaliknya, kegagalan validasi akan langsung menghasilkan respons kesalahan dengan format yang konsisten dan informatif, sehingga integritas sistem tetap terjaga.



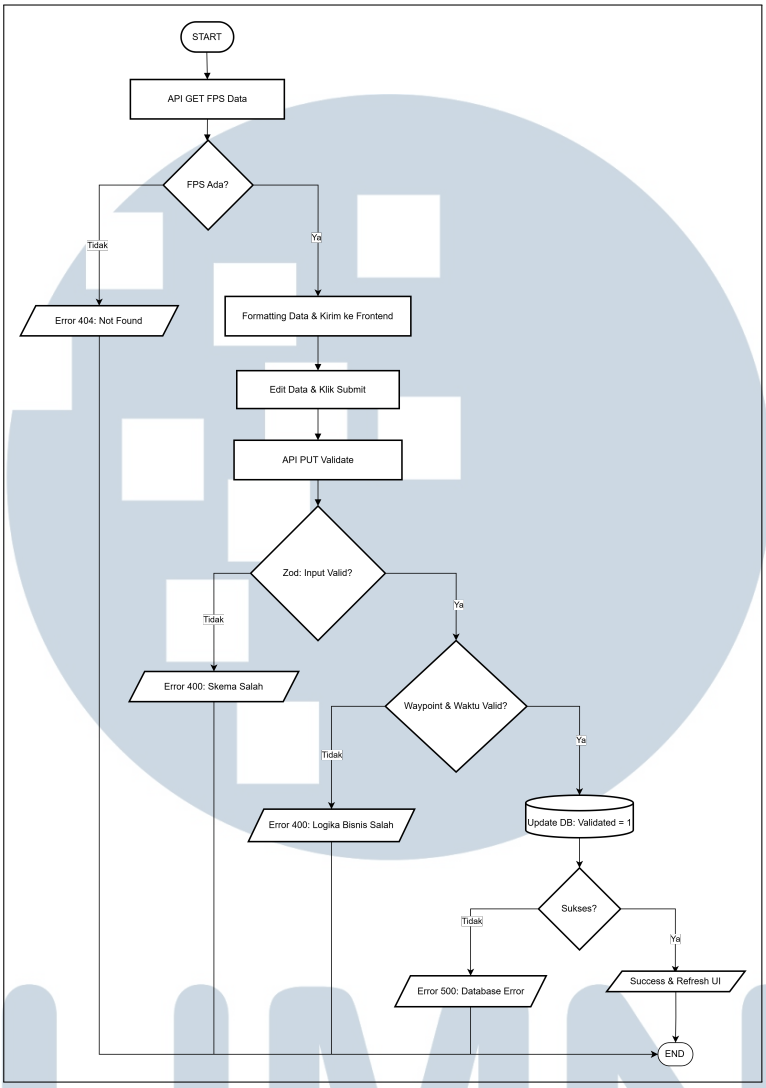
Gambar 3.3. *Flowchart* Proses validasi

3. Business Logic

- FPS Validation Flow

  Proses validasi Flight Plan System (FPS) dirancang agar setiap perubahan data yang dilakukan pengguna melalui antarmuka sistem selalu melewati tahapan validasi berlapis, mulai dari validasi input berbasis skema hingga validasi aturan bisnis operasional. Alur ini melibatkan interaksi antara frontend dan backend melalui endpoint REST API, dengan pemanfaatan Zod untuk validasi data serta Prisma untuk operasi pembaruan ke basis data MySQL

Proses validasi Flight Plan Summary (FPS) dimulai saat pengguna menekan tombol “Validate”, sehingga sistem memuat data FPS melalui permintaan GET, memeriksa keberadaan dan memformat data (termasuk konversi waktu dan status boolean), lalu menampilkan formulir yang dapat diedit sebagian oleh pengguna; setelah pengguna mengirim perubahan melalui permintaan PUT, sistem melakukan validasi skema dan aturan bisnis (cek waypoint, urutan waktu, konsistensi DOF, dan referensi tipe pesawat), kemudian jika semua validasi terpenuhi, data FPS diperbarui di basis data dengan menandai status Validated = ”1” dan mengembalikan respons sukses yang direspons frontend dengan pembaruan tampilan, sedangkan jika terjadi kesalahan validasi atau pembaruan, sistem mengembalikan pesan error agar pengguna dapat memperbaiki input atau mengetahui adanya gangguan internal

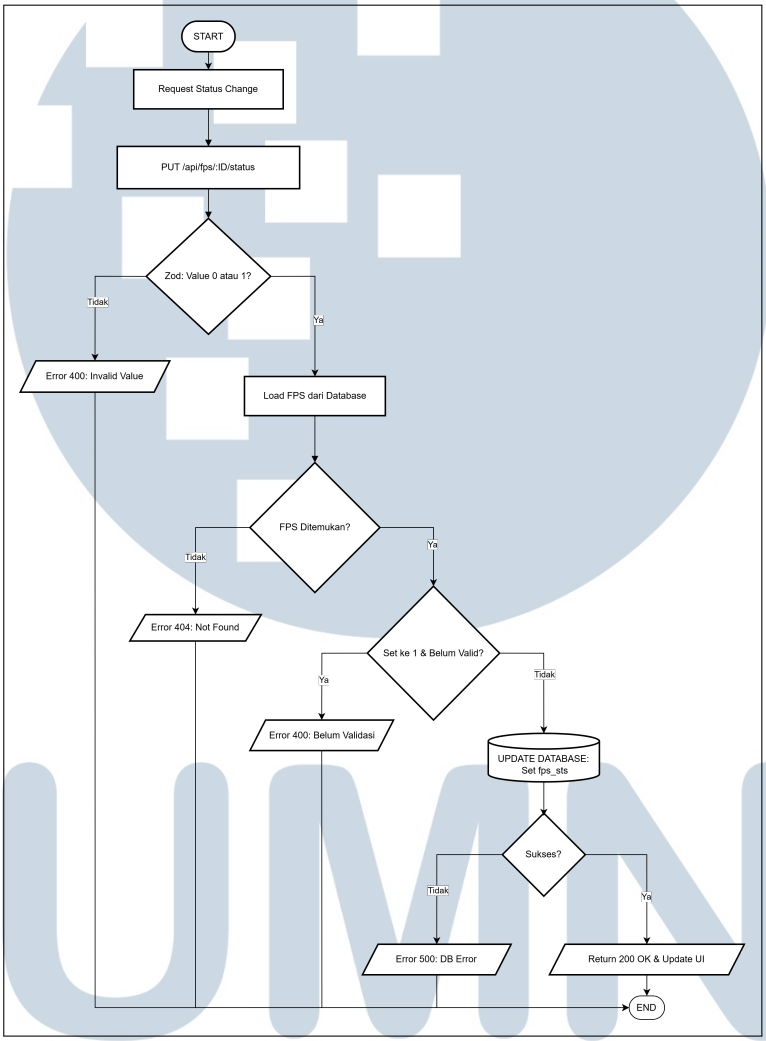


Gambar 3.4. *Flowchart* Proses validasi pada Flight Plan System

- FPS Status Management

  Proses pengelolaan status Flight Plan System (FPS) memungkinkan pergantian status antara ”0” (inactive) dan ”1” (active) melalui endpoint PUT /api/fps/:ID/status dengan payload ”fps_sts”: ”1”, di mana backend pertama-tama melakukan validasi Zod untuk memastikan nilai fps_sts hanya ”0” atau ”1”, memeriksa keberadaan FPS berdasarkan ID di basis data, serta melakukan pemeriksaan opsional bahwa FPS harus tervalidasi (Validated = ”1”) sebelum diaktifkan; jika semua kondisi terpenuhi, sistem memperbarui tabel all_ovf menggunakan Prisma dengan mengubah fps_sts, mencatat fps_updater dan fps_time_updated, kemudian mengembalikan respons sukses (200 OK) yang memicu

frontend untuk memperbarui tampilan UI (menampilkan status hijau/aktif) dan notifikasi keberhasilan, sedangkan kegagalan validasi, FPS tidak ditemukan, atau error basis data menghasilkan respons error (400/404/500) dengan pesan kesalahan yang sesuai.

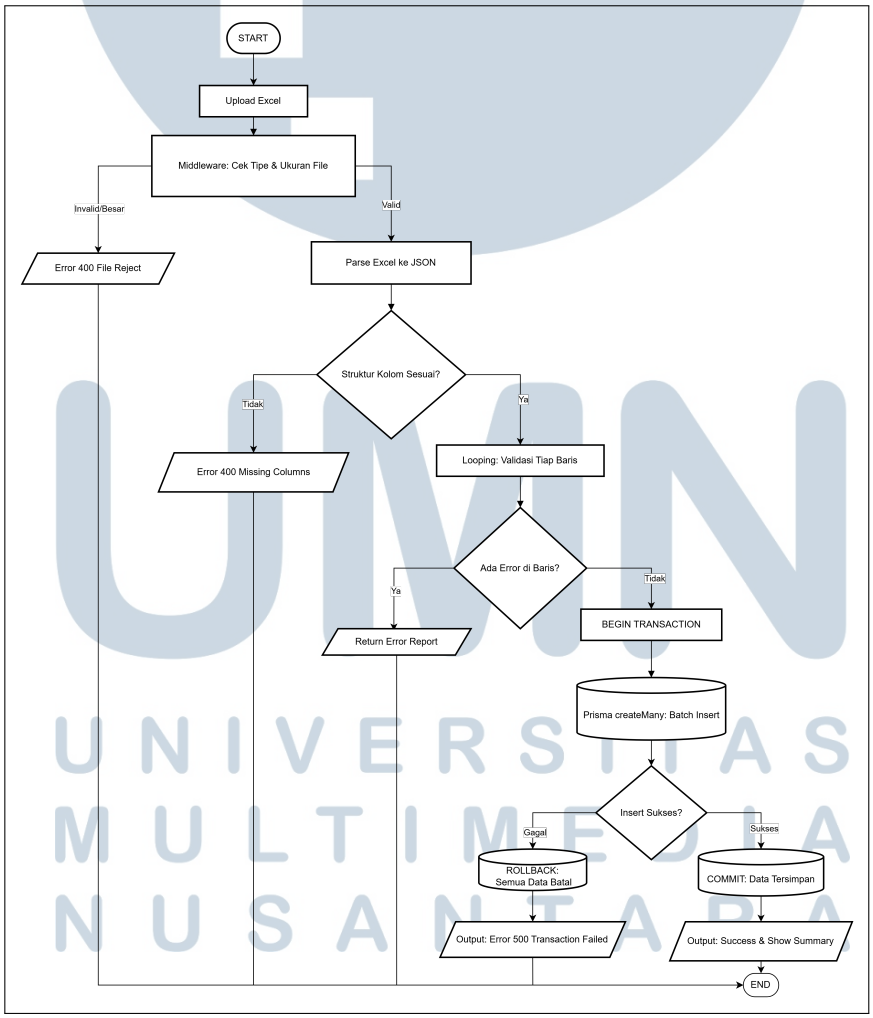


Gambar 3.5. *Flowchart* Proses pengelolaan status Flight Plan System

- A-CARE Bulk Upload

  Proses unggah data A-CARE melalui endpoint POST /api/acare/upload mencakup beberapa tahap utama, yaitu pengiriman file Excel (.xlsx/.xls, maksimal 10 MB) dari frontend menggunakan multipart/form-data, validasi tipe dan ukuran file oleh Multer, pemrosesan file dengan pustaka XLSX untuk mengonversi lembar pertama menjadi JSON (maksimal 10.000 baris), pemeriksaan

struktur kolom wajib, serta validasi per baris yang mencakup panjang CallSign maksimal 10 karakter, ADEP/ADES tepat 4 karakter, format DOF sebagai tanggal valid, Time_In/Time_Out sebagai HH:MM:SS, ARR_DEP bernilai ’A’ atau ’D’, DOM_INT bernilai ’DOM’ atau ’INT’, dan nilai numerik pada bidang biaya; apabila terdapat kesalahan, sistem mengembalikan respons 400 beserta daftar error per baris, sedangkan jika semua data valid, Prisma menjalankan transaksi createMany secara atomik untuk menyisipkan seluruh baris ke tabel data_acare dengan konversi tipe DOF menjadi Date, sehingga keberhasilan menghasilkan respons 200 OK yang mencantumkan jumlah record terinsert dan durasi proses, sementara kegagalan memicu rollback otomatis guna menjaga konsistensi basis data.



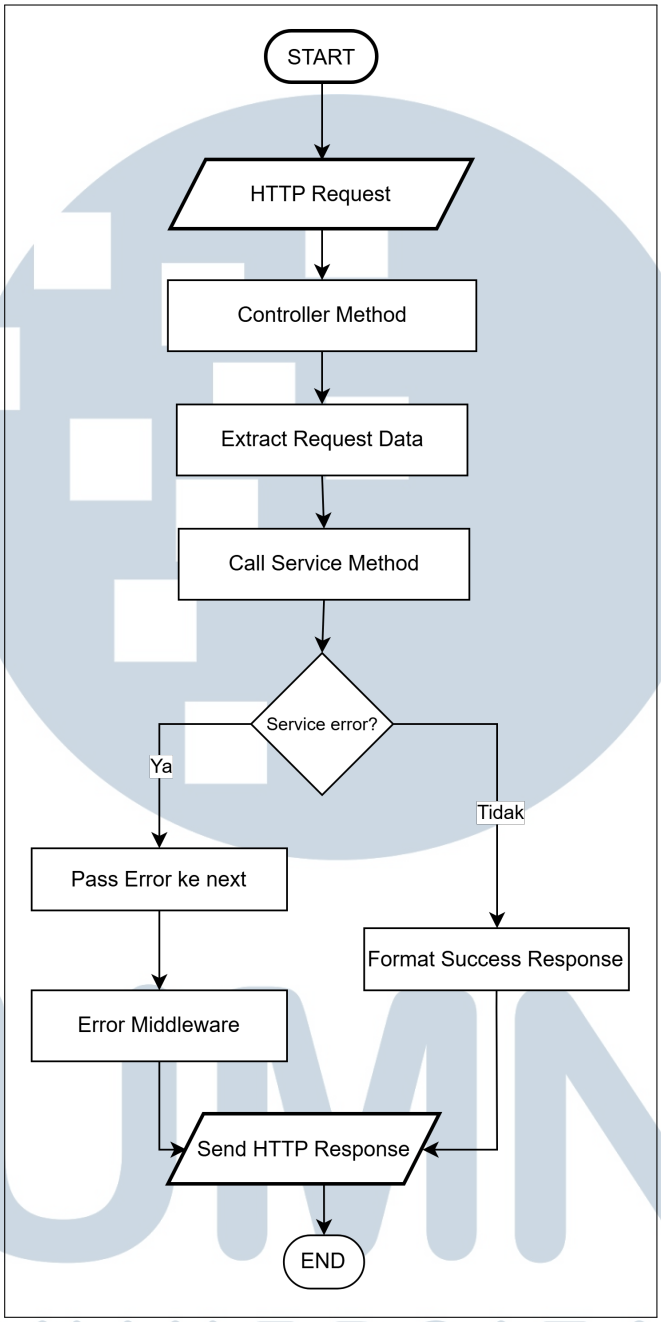
Gambar 3.6. Proses unggah data A-CARE

3.3.4 Implementasi Fitur

A Implementasi Layer Architecture

1. Controller Layer

   Controller Layer merupakan lapisan teratas dalam arsitektur sistem yang berinteraksi langsung dengan klien melalui protokol HTTP. Lapisan ini bertanggung jawab menangani komunikasi HTTP dengan menerima permintaan, mengekstraksi data dari parameter, query, dan body, serta memanggil metode yang sesuai pada Service Layer [8]. Hasil pemrosesan kemudian diformat menjadi respons JSON yang konsisten dan dikirimkan dengan kode status HTTP yang sesuai. Kesalahan yang terjadi selama proses diteruskan ke error middleware untuk penanganan terpusat. Implementasi Controller pada HAADES Backend menerapkan prinsip thin controller, di mana Controller tidak mengandung logika bisnis. Seluruh logika bisnis, validasi, dan perhitungan didelegasikan kepada Service Layer, sehingga Controller hanya berfokus pada pemrosesan permintaan dan respons [8]. Pendekatan ini meningkatkan keterujian, keterbacaan, dan kemudahan pemeliharaan sistem.

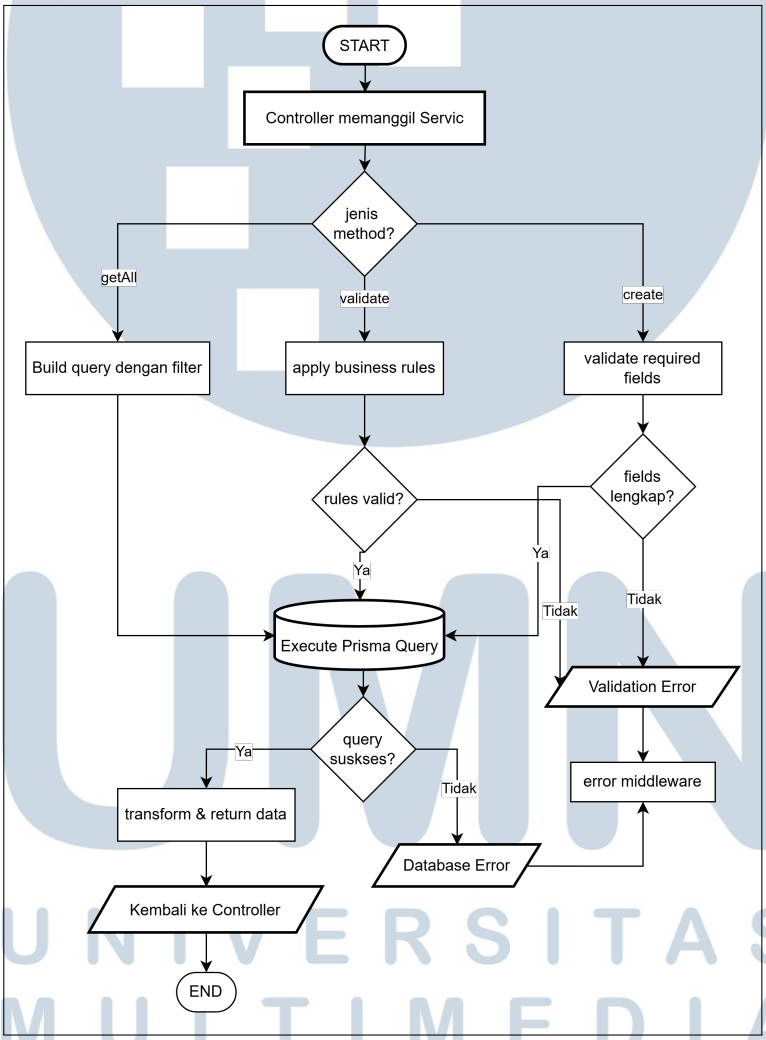


Gambar 3.7. *Flowchart* Proses pada controller layer

2. Service Layer (Business Logic)

   Service Layer merupakan komponen inti dalam aplikasi HAADES Backend yang mengimplementasikan logika bisnis dan aturan operasional sistem. Lapisan ini berperan sebagai penghubung antara Controller yang menangani komunikasi HTTP dan Model yang mengakses basis data. Service Layer bertanggung jawab atas validasi aturan bisnis, pengoordinasian transaksi

basis data, transformasi data antara format penyimpanan dan aplikasi, serta penanganan kesalahan secara terstruktur. Pemisahan Service Layer dari Controller mendukung keterpisahan tanggung jawab dan meningkatkan fleksibilitas arsitektur. Logika bisnis dapat digunakan kembali dalam berbagai konteks, diuji secara independen tanpa ketergantungan pada web server, serta memudahkan pemeliharaan dan pengembangan sistem. Pendekatan ini juga memungkinkan perubahan transport layer, seperti dari REST ke GraphQL, tanpa memengaruhi logika bisnis yang ada.

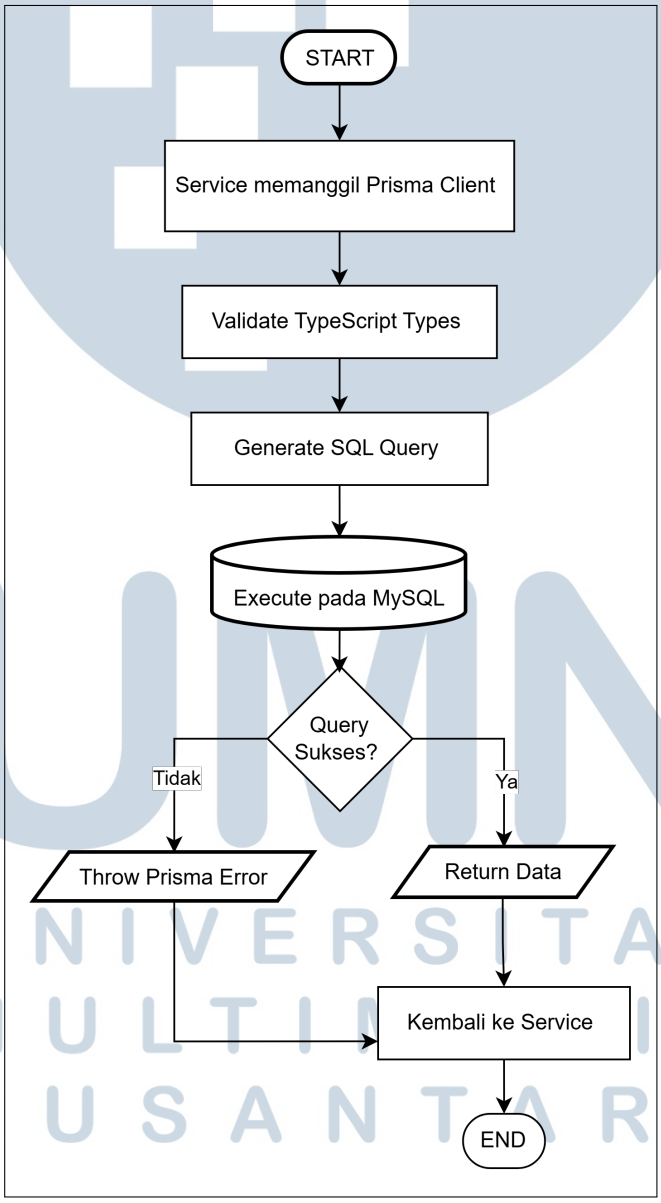


Gambar 3.8. *Flowchart* Proses pada Service layer

3. Model Layer (Data Access)

Model Layer merupakan lapisan terbawah dalam arsitektur sistem yang berinteraksi langsung dengan basis data MySQL dengan memanfaatkan

Prisma Object-Relational Mapping (ORM) untuk menyediakan akses data yang bersifat type-safe. Lapisan ini mengabstraksi kompleksitas SQL dan operasi spesifik basis data sehingga Service Layer tidak bergantung pada detail implementasi akses data. Prisma dipilih karena kemampuannya menghasilkan tipe data TypeScript secara otomatis dari skema basis data, mendukung auto-completion pada IDE, menyediakan pesan kesalahan yang informatif, serta dilengkapi migration tools dan Prisma Studio untuk pengelolaan skema dan data selama pengembangan.



Gambar 3.9. *Flowchart* Proses pada Model layer

B Implementasi Middleware

1. Error Middleware

   Error middleware merupakan komponen sentral dalam penanganan kesalahan di HAADES Backend. Middleware ini dirancang untuk menangani berbagai jenis kesalahan yang mungkin terjadi selama pemrosesan permintaan dan mengkonversinya menjadi respons HTTP yang konsisten dan informatif.

```
FUNCTION globalErrorHandler(error, request, response):
    LOG(error) // Mencatat error untuk audit internal

    IF error IS ResponseError:
        RETURN RESPONSE(
            status  = error.code,
            message = error.message
        )

    IF error IS ZodValidationError:
        RETURN RESPONSE(
            status  = 400,
            message = "Integritas Data Gagal",
            details = error.list
        )

    IF error IS DatabaseError:
        IF code IS "UNIQUE_VIOLATION":
            RETURN RESPONSE(
                status  = 400,
                message = "Data Sudah Terdaftar"
            )
        IF code IS "NOT_FOUND":
            RETURN RESPONSE(
                status  = 404,
                message = "Data Tidak Ada"
            )

    // Penanganan default untuk kesalahan sistem
    RETURN RESPONSE(
        status  = 500,
        message = "Gangguan Server Internal"
    )

```

Kode 3.5: Pseudocode Error Handler

Error middleware ini menangani tiga kategori utama kesalahan dengan pendekatan yang berbeda untuk masing-masing. Untuk ResponseError yang merupakan kesalahan custom yang dilempar oleh Service Layer, middleware menggunakan kode status dan pesan yang sudah ditentukan dalam objek error. Untuk ZodError yang terjadi saat validasi gagal, middleware memformat

daftar kesalahan menjadi struktur yang mudah dipahami client. Untuk kesalahan Prisma seperti constraint violation atau data tidak ditemukan, middleware menerjemahkan kode error Prisma menjadi pesan yang user-friendly. Untuk kesalahan lain yang tidak terduga, respons internal server error generik dikembalikan, dengan detail error hanya ditampilkan pada mode pengembangan untuk keamanan.

2. Upload Middleware

   Upload middleware menggunakan Multer untuk menangani upload file dengan konfigurasi yang aman dan terbatas.

```
DEFINE ConfigUploadFile:
    LOKASI_SIMPAN      : MEMORI_BUFFER
    MAKSIMAL_UKURAN    : 10MB

    FILTER_TIPE_FILE:
        HANYA IZINKAN : [".xlsx", ".xls", ".csv"]
        JIKA LAIN     : TOLAK DENGAN PESAN
                        "Format File Tidak Didukung"

```

Kode 3.6: Pseudocode Konfigurasi Upload File untuk Validasi dan Pembatasan Berkas

Konfigurasi upload middleware dirancang dengan mempertimbangkan keamanan dan efisiensi. File disimpan dalam memori sebagai Buffer untuk pemrosesan langsung tanpa perlu menulis ke disk terlebih dahulu, mengurangi I/O overhead. Batasan ukuran file diterapkan untuk mencegah upload file yang terlalu besar yang dapat menghabiskan memori server. File filter memvalidasi tipe MIME file yang di-upload untuk memastikan hanya file Excel dan CSV yang diterima, mencegah upload file eksekutabel atau tipe file berbahaya lainnya. Pendekatan ini memberikan keseimbangan antara keamanan, performa, dan kemudahan penggunaan.

C Implementasi Routing

Routing merupakan mekanisme untuk mendefinisikan bagaimana aplikasi merespons permintaan client ke endpoint tertentu dengan metode HTTP spesifik [9]. HAADES Backend menggunakan Express Router untuk mengorganisir routes secara modular, dengan memisahkan routes berdasarkan entitas atau fitur untuk meningkatkan maintainability dan readability kode. Pendekatan ini memastikan bahwa setiap request dari client diarahkan ke Controller yang tepat setelah melewati lapisan keamanan middleware (seperti validasi skema dan penanganan upload file).

```
        // 1. Inisialisasi Router Utama
        ROUTER_API = CREATE_EXPRESS_ROUTER()
        // 2. Definisi Pemetaan Endpoint (Mapping)
        OPERASI RegistrasiRute(jalur, metode, validator, kontroler):
            IF validator IS_SET:
                ROUTER_API[metode](<jalur, middleware(validator>), kontroler)
            ELSE:
                ROUTER_API[metode](<jalur, kontroler>)
        // Contoh Implementasi pada Modul FPS
        RegistrasiRute("/fps", "GET", NULL, FPSController.list)
        RegistrasiRute("/fps", "POST", FPSValidation.CREATE, FPSController.create)
        RegistrasiRute("/fps/:id/validate", "PUT", FPSValidation.VALIDATION,
    FPSController.validate)
        // Contoh Implementasi Upload (Multipart)
        RegistrasiRute("/acare/upload", "POST", UPLOAD_MIDDLEWARE, AcareController
    .upload)
        // 3. Ekspor Router ke Server Utama
        EXPORT ROUTER_API

```

Kode 3.7: Pseudocode Rute pada Modul Manajemen Penerbangan (FPS)

Dengan struktur pengkodean rute seperti di atas, sistem mencapai pemisahan tugas yang sangat baik. Router bertindak sebagai ”polisi lalu lintas” yang bertugas melakukan pencocokan alamat URL, memicu validasi data di gerbang masuk (middleware), dan akhirnya menyerahkan eksekusi ke lapisan logika bisnis di Controller [9]. Penggunaan prefix /api di file main.ts memberikan pemisahan yang jelas antara resource backend dengan rute-rute aplikasi lainnya, sementara penggunaan rute dinamis (seperti /:id) memungkinkan sistem untuk menangani manipulasi data spesifik secara efisien. Prosedur penanganan rute terakhir (404 handler) juga memastikan bahwa request ke endpoint yang tidak terdaftar akan ditolak dengan respons JSON yang standar, menjaga konsistensi pengalaman pengguna API.

D Implementasi Fitur Khusus

1. Fitur Bulk Upload Data A-CARE (Excel Processing)

   Fitur bulk upload memungkinkan pengunggahan data A-CARE dalam jumlah besar melalui file Excel dengan menerapkan validasi bertahap dan transaksi basis data untuk menjaga integritas data. Proses dimulai dengan pembacaan file menggunakan pustaka XLSX dan konversi sheet pertama menjadi array objek, kemudian setiap baris divalidasi melalui pemeriksaan struktur dan referential integrity. Seluruh proses validasi dan penyimpanan data dibungkus dalam transaksi Prisma untuk menjamin atomisitas, di mana data yang valid

tetap diproses meskipun terdapat baris yang tidak valid. Hasil pemrosesan mencakup jumlah data yang berhasil disimpan, total data yang diunggah, serta daftar kesalahan yang teridentifikasi.

```
FUNCTION processBulkUpload(fileBuffer):
    TRY:
        // Tahap 1: Parsing Data
        workbook = READ_EXCEL(fileBuffer)
        data_rows = CONVERT_TO_JSON(workbook.Sheet[0])

        IF data_rows.length == 0:
            THROW ERROR "File Kosong"

        // Tahap 2: Database Transaction
        BEGIN TRANSACTION:
            FOR EACH row IN data_rows:
                // Validasi Struktur Dasar
                VALIDATE_SCHEMA(row)

                // Validasi Integritas Referensial
                IF NOT EXISTS(point_in_db WHERE name == row.Pnt_In):
                    COLLECT_ERROR(row_index, "Waypoint Not Found")
                    CONTINUE

                ADD_TO_VALID_LIST(row)

            // Tahap 3: Batch Insertion
            IF valid_list.length > 0:
                INSERT_MANY_TO_DB(valid_list, skipDuplicates=TRUE)
            ELSE:
                ROLLBACK TRANSACTION
                THROW ERROR "Tidak ada data valid"

        COMMIT TRANSACTION
        RETURN { inserted: count, errors: error_list }

    CATCH (error):
        LOG(error)
        THROW error
```

Kode 3.8: Pseudocode Proses Bulk Upload Data dengan Validasi dan Transaksi Basis Data

Dengan menggunakan logika di atas, sistem menjamin bahwa data hanya akan tersimpan jika memenuhi kriteria validasi teknis dan bisnis. Library XLSX digunakan untuk membedah data biner, sementara Prisma $transaction memastikan bahwa jika terjadi kegagalan sistemik, status database akan dikembalikan ke kondisi semula (rollback).

2. Fitur Impor Data CSV

Selain mendukung format Excel, sistem HAADES juga menyediakan fitur impor data melalui file CSV (Comma-Separated Values). Fitur ini ditujukan untuk interoperabilitas dengan sistem warisan (legacy system) yang sering kali mengekspor data dalam format teks sederhana. Proses impor CSV dirancang untuk menangani ribuan baris data secara efisien dengan beban memori yang minimal melalui teknik streaming atau line-by-line parsing.

```
FUNCTION importCSVFile(fileBuffer):
    TRY:
        // Tahap 1: Membaca dan Mengonversi
        raw_text = DECODE_BOM_AND_READ(fileBuffer)
        csv_rows = PARSE_CSV_TO_OBJECTS(raw_text, delimiter=",")

        IF csv_rows.length == 0:
            THROW ERROR "Format CSV Tidak Valid"

        // Tahap 2: Validasi dan Sanitasi
        valid_items = []
        errors = []

        FOR EACH row IN csv_rows:
            IF VALIDATE_COLUMNS(row, expected=["CallSign", "DOF", "ADEP"]):
                // Membersihkan spasi atau karakter ilegal
                sanitized_row = TRIM_VALUES(row)
                valid_items.PUSH(sanitized_row)
            ELSE:
                errors.PUSH({ row_number: row.id, detail: "Kolom Tidak Lengkap" })

        // Tahap 3: Penyimpanan Massal
        IF valid_items.length > 0:
            PERFORM_DATABASE_UPSERT(valid_items)

        RETURN {
            status: "Success",
            processed: valid_items.length,
            ignored: errors.length
        }

    CATCH (exception):
        REPORT_CRITICAL_FAILURE(exception)
        RETHROW exception
```

Kode 3.9: Pseudocode Import Berkas CSV dengan Validasi

Implementasi fitur ini memastikan bahwa data dari berbagai sumber dapat diseragamkan (standardized) sebelum masuk ke repositori basis data. Mekanisme pembersihan data (sanitization) pada tahap kedua sangat penting untuk mencegah masuknya karakter teks yang rusak atau tidak terlihat yang

sering ditemukan pada file CSV manual. Hal ini menjamin bahwa data yang diimpor tetap konsisten dengan skema Zod yang telah ditetapkan pada lapisan aplikasi lainnya.

3. Workflow Verifikasi Prosedural FPS

   Fitur ini merupakan inti dari fungsionalitas validasi operasional. Sebelum data Flight Plan (FPS) masuk ke tahap penagihan, petugas harus memverifikasi kesesuaian titik masuk/keluar (Waypoints) dan waktu tempuh. Sistem akan mengubah status data menjadi Validated hanya jika parameter teknis rute telah terpenuhi

```
FUNCTION verifyFlightPlan(fps_id, validation_data):
    // 1. Mengambil data Flight Plan System (FPS) asli
    current_fps = FIND_FPS_BY_ID(fps_id)

    // 2. Validasi Aturan Bisnis (Service Level)
    IF validation_data.Time_Out <= validation_data.Time_In:
        THROW ERROR "Waktu keluar tidak boleh mendahului waktu masuk"

    IF NOT IS_VALID_AIRPORT(current_fps.ADEP) OR
       NOT IS_VALID_AIRPORT(current_fps.ADES):
        THROW ERROR "Kode Bandara ICAO tidak terdaftar"

    // 3. Memperbarui Status dan Menyimpan Perubahan
    UPDATE_DB(fps_table):
        SET data = validation_data,
        SET Validated = '1'
        WHERE ID == fps_id

    RETURN SUCCESS_MESSAGE

```

Kode 3.10: Pseudocode Verifikasi Flight Plan pada Service Layer

   Workflow ini memastikan adanya pemisahan antara data mentah yang di-upload dan data yang telah melewati sensor operasional manusia. Dengan mengubah status Validated menjadi '1', sistem secara otomatis membuka akses bagi modul Billing Engine untuk menarik data tersebut ke dalam laporan keuangan.

4. Sistem Audit Log Otomatis

   Dalam sistem navigasi udara, akuntabilitas setiap perubahan data sangat krusial. Fitur Audit Log mencatat setiap aksi manipulasi data (Create, Update, Delete) yang dilakukan oleh pengguna. Informasi yang dicatat

meliputi siapa pelakunya, kapan dilakukan, modul apa yang diubah, serta data apa yang dikirimkan.

```
FUNCTION createAuditLog(action, module, payload, user_id):
    // Menyiapkan entri log
    log_entry = {
        timestamp     : GET_CURRENT_TIME(),
        user          : user_id,
        action_type   : action,         // Contoh: 'UPDATE_FPS'
        module_name   : module,         // Contoh: 'FPS_MANAGEMENT'
        data_snapshot : STRINGIFY(payload),
        client_ip     : GET_REQUEST_IP()
    }

    // Menyimpan data ke tabel log_system secara asinkron
    ASYNC_SAVE_TO_DB(log_entry)

```

Kode 3.11: Pseudocode Proses Pencatatan Audit Log pada Sistem Backend

Log ini diimplementasikan pada tingkat Service Layer atau melalui Global Middleware. Pencatatan asinkron digunakan agar proses penyimpanan log tidak menambah waktu respons (latency) pada pengguna akhir, sehingga performa aplikasi tetap optimal meskipun sedang menangani lalu lintas data yang padat.

E CORS Configuration

Cross-Origin Resource Sharing (CORS) merupakan mekanisme keamanan browser yang membatasi permintaan HTTP dari origin yang berbeda. Konfigurasi CORS yang tepat sangat penting untuk memungkinkan frontend berkomunikasi dengan backend sambil tetap menjaga keamanan [10].

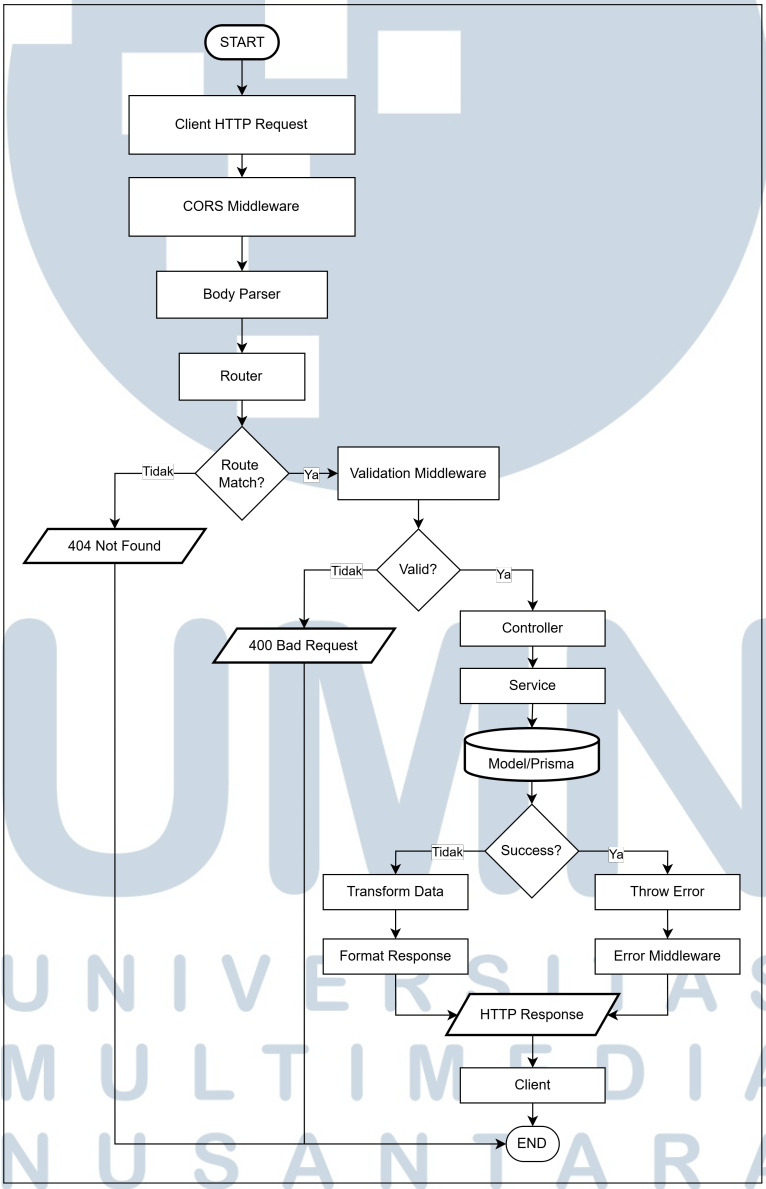
```
import cors from 'cors';

export const corsMiddleware = cors({
  origin: process.env.CORS_ORIGIN || 'http://localhost:5173',
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH'],
  allowedHeaders: ['Content-Type', 'Authorization'],
  exposedHeaders: ['Content-Range', 'X-Content-Range'],
  maxAge: 86400, // Cache preflight 24 jam
});
```

Kode 3.12: Contoh code CORS Configuration

Konfigurasi CORS dirancang dengan mempertimbangkan keamanan dan fleksibilitas dengan menggunakan variabel environment untuk menentukan

origin yang diizinkan, memungkinkan konfigurasi berbeda untuk pengembangan dan produksi. Opsi credentials: true mengizinkan pengiriman cookies dan authorization headers untuk autentikasi. Metode HTTP yang diizinkan dibatasi hanya pada yang diperlukan oleh API. Headers yang dapat dikirim dan diterima didefinisikan secara eksplisit untuk keamanan, dan maxAge diatur untuk meng-cache hasil preflight request selama 24 jam, mengurangi jumlah permintaan options yang tidak perlu.



Gambar 3.10. *Flowchart* proses CORS

Flowchart di atas menggambarkan alur lengkap dari permintaan HTTP

hingga respons dikirim kembali ke client. Permintaan pertama kali melewati CORS middleware untuk verifikasi origin, kemudian body parser untuk mengkonversi data JSON atau URL-encoded. Router mencocokkan path dan metode dengan route yang terdaftar. Jika tidak ada yang cocok, respons 404 dikembalikan. Jika cocok, validation middleware memvalidasi input menggunakan skema Zod. Data yang valid kemudian diproses oleh Controller yang memanggil Service untuk logika bisnis, yang pada gilirannya memanggil Model untuk operasi basis data. Hasil sukses ditransformasi dan diformat, sementara kesalahan ditangani oleh error middleware sebelum respons dikirim ke client.

3.3.5 Pengujian Fitur

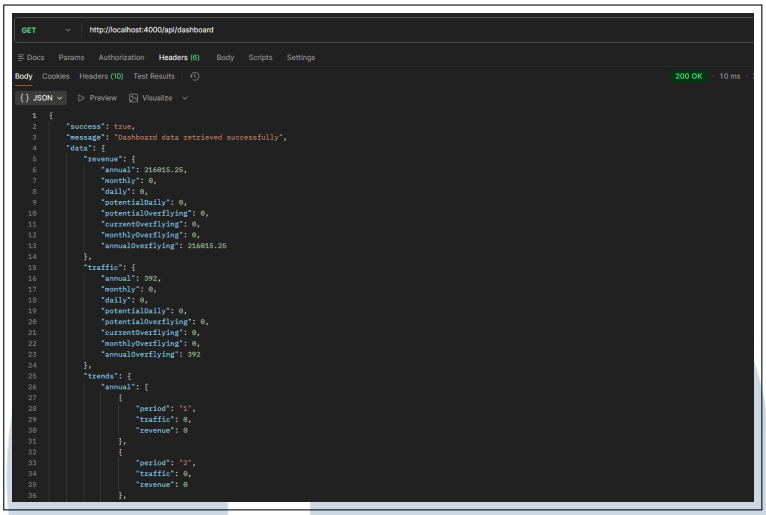
Pengujian merupakan tahap penting dalam pengembangan HAADES Backend untuk memastikan aplikasi berfungsi sesuai spesifikasi dan memiliki keandalan yang tinggi. Strategi pengujian diterapkan secara multi-layer dengan mencakup pengujian API, validasi input, integrasi antarkomponen (Controller, Service, dan Model), penanganan kesalahan, serta performa sistem. Setiap pengujian dirancang menggunakan skenario happy path, edge case, dan kasus negatif guna memastikan aplikasi mampu menangani berbagai kondisi operasional. Hasil pengujian didokumentasikan secara sistematis untuk mendukung evaluasi dan peningkatan kualitas sistem.

A Pengujian API dengan Postman

Pengujian API menggunakan Postman dilakukan dengan pendekatan sistematis untuk memastikan setiap endpoint merespons sesuai kontrak API yang telah didefinisikan. Pengujian diorganisir ke dalam sebuah Koleksi (Collection) yang berisi folder-folder untuk setiap modul fungsional. Strategi ini memungkinkan pengujian mandiri per fitur maupun pengujian alur kerja (workflow) yang melibatkan beberapa modul.

1. Modul Dashboard

   Pengujian pada modul Dashboard dilakukan untuk memverifikasi fungsionalitas agregasi data real-time. Server mengumpulkan statistik pendapatan (revenue), lalu lintas penerbangan (traffic), dan tren periode harian, bulanan, hingga tahunan dari tabel all_ovf dan addovf.

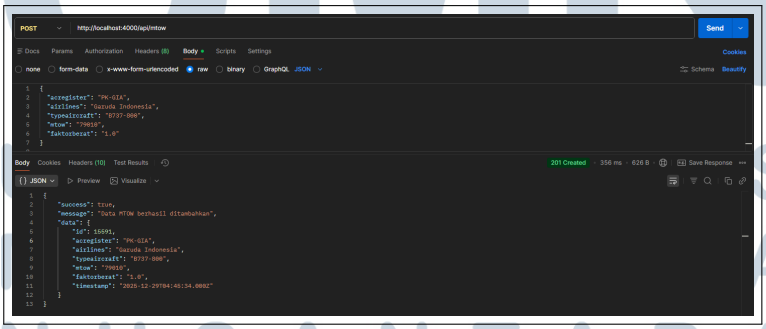


Gambar 3.11. Hasil Pengujian Endpoint Modul Dashboard Menggunakan Postman

Hasil pengujian ini menunjukkan struktur data yang komprehensif mencakup metrik finansial dan operasional. Kehadiran objek revenue dan traffic yang terbagi ke dalam berbagai periode waktu membuktikan bahwa fungsi kalkulasi asinkron di backend berjalan dengan presisi, memungkinkan visualisasi data yang akurat pada grafik dashboard.

2. MTOW

Master Data adalah referensi utama sistem. Pengujian ini memastikan fungsionalitas CRUD standar berjalan baik. Akurasi data di sini sangat krusial karena akan digunakan sebagai basis perhitungan oleh modul FPS dan A-CARE.



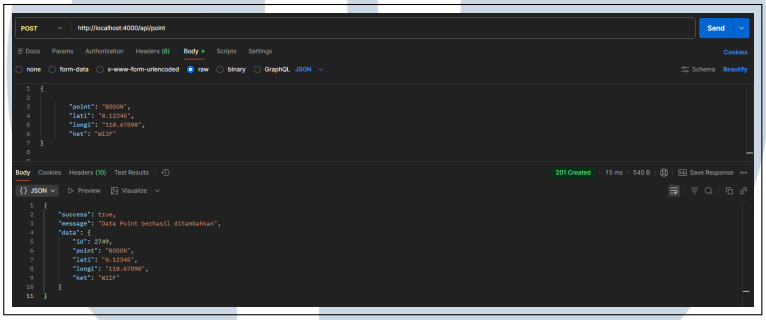
Gambar 3.12. Hasil Pengujian Endpoint Modul MTOW Menggunakan Postman

Pesan sukses (201 Created) memvalidasi bahwa skema data telah sesuai dan berhasil disimpan ke MySQL. Penyimpanan ID ke variabel lingkungan

memastikan alur pengujian otomatis dapat berlanjut ke tahap update atau delete.

3. Point

   Modul Point menyimpan koordinat titik navigasi di wilayah udara Indonesia. Pengujian ini memverifikasi kemampuan sistem dalam mengelola daftar waypoints (seperti BOSON, PUMEG) yang menjadi acuan validasi rute pada Flight Plan.

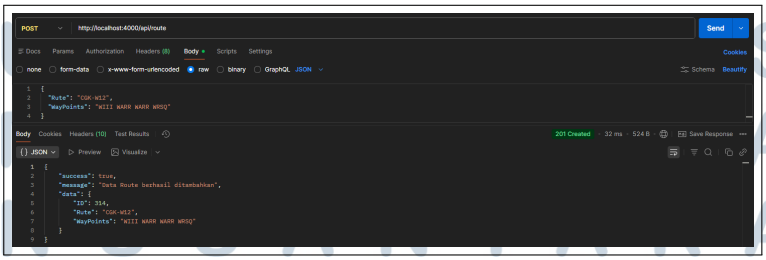


Gambar 3.13. Hasil Pengujian Endpoint Modul Point Menggunakan Postman

Keberhasilan input data Point memastikan bahwa Service Layer memiliki database referensi geospasial yang mutakhir untuk memverifikasi apakah sebuah pesawat benar-benar melintasi titik yang dilaporkan.

4. Route

   Modul Route mengelola segmen jalur penerbangan yang menghubungkan antar titik (Point). Pengujian ini memastikan sistem dapat mendefinisikan rute standar untuk mempermudah deteksi anomali pada rencana terbang yang diajukan maskapai.



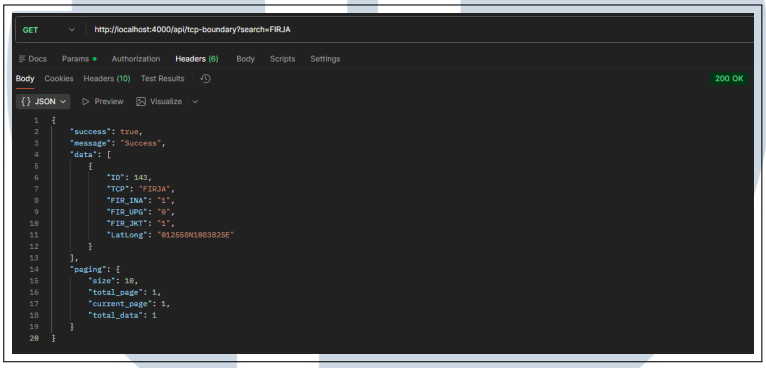
Gambar 3.14. Hasil Pengujian Endpoint Modul Route Menggunakan Postman

Input rute yang valid memungkinkan sistem melakukan perhitungan jarak

otomatis secara presisi, yang berdampak langsung pada nilai tagihan jasa navigasi udara.

5. TCP

   TCP adalah titik kendali terminal udara. Pengujian dilakukan untuk memastikan sinkronisasi data antar titik kontrol wilayah udara (FIR) terpelihara, sehingga proses hand-over navigasi tercatat dengan benar di sistem audit.

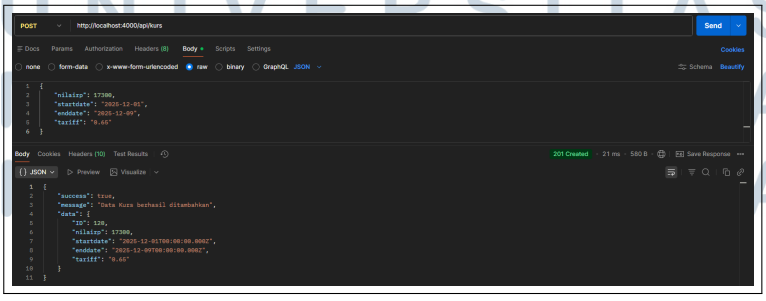


Gambar 3.15. Hasil Pengujian Endpoint Modul TCP Menggunakan Postman

Hasil pencarian TCP yang akurat membuktikan bahwa sistem filter pada repositori database bekerja optimal, memudahkan petugas menemukan titik kontrol yang relevan secara cepat.

6. Kurs

   Karena penagihan navigasi internasional sering menggunakan mata uang asing (USD), modul Kurs sangat krusial. Pengujian ini memverifikasi bahwa sistem memiliki nilai tukar rupiah (nilairp) dan tarif (tariff) yang berlaku pada rentang waktu tertentu (startdate - enddate).

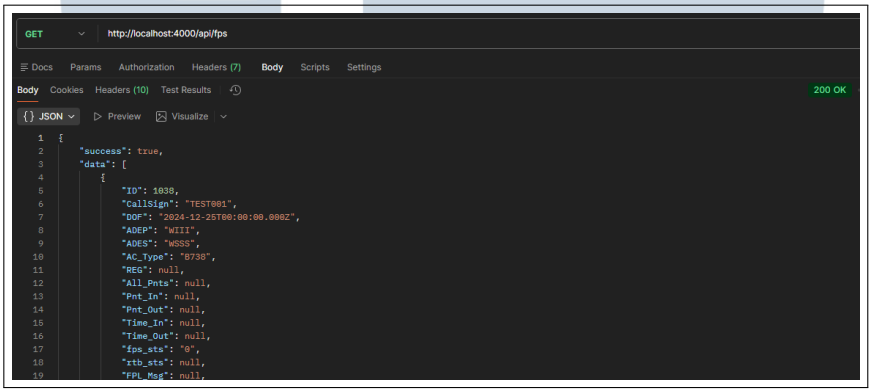


Gambar 3.16. Hasil Pengujian Endpoint Modul Kurs Menggunakan Postman

Validasi pada modul Kurs menjamin bahwa sistem memiliki referensi nilai tukar dan tarif yang sah. Data nilairp digunakan untuk konversi biaya jasa navigasi ke Rupiah, sementara tariff menentukan nilai dasar unit rate yang berlaku pada periode tersebut sesuai kebijakan otoritas udara.

7. Modul FPS

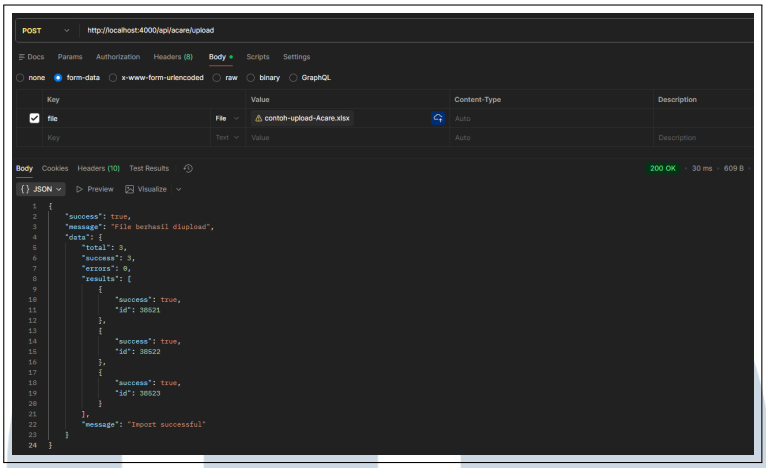
   FPS adalah inti operasional sistem HAADES. Pengujian dilakukan melalui dua tahap utama: verifikasi data teknis rute (/validate) dan aktivasi status operasional (/status). Endpoint mengizinkan petugas untuk melengkapi data yang belum terisi seperti Point In, Point Out, dan registrasi pesawat.



Gambar 3.17. Hasil Pengujian Endpoint Modul FPS

8. Modul A-CARE (Data Import & Management)

   Modul A-CARE menangani data eksternal dalam jumlah besar. Pengujian ini memverifikasi proses unggah file Excel agar dapat terurai (parsing) menjadi entri database secara otomatis tanpa kegagalan memori.

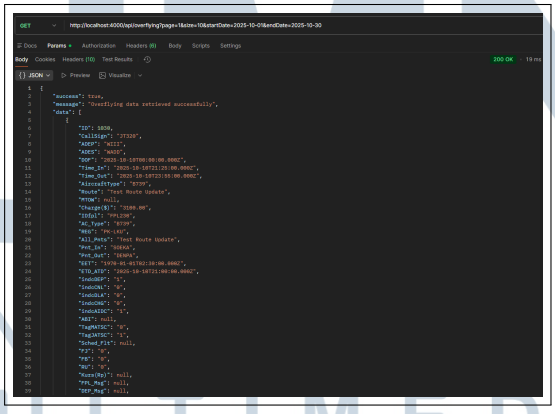


Gambar 3.18. Hasil Pengujian Endpoint Modul A-Care Menggunakan Postman

Munculnya jumlah data yang berhasil diinput (inserted 0) mengonfirmasi bahwa library pemroses Excel bekerja sinkron dengan repositori data, menjamin efisiensi dalam pengelolaan ribuan data penerbangan.

9. Modul Overflying

Modul Overflying menyajikan daftar seluruh data navigasi yang terekam pada tabel all_ovf. Pengujian difokuskan pada kemampuan filter database yang kompleks, mencakup kriteria operasional seperti StatusDelete = 0, TagJATSC = 1, serta rentang waktu masuk dan keluar wilayah udara.

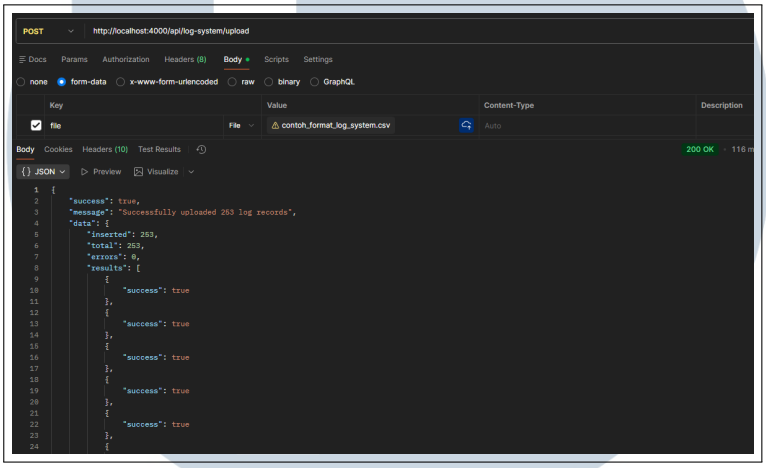


Gambar 3.19. Hasil Pengujian Endpoint Modul Overflying Menggunakan Postman

Respons yang menyertakan detail teknis penerbangan beserta data paginasi mengonfirmasi bahwa Query Raw Unsafe di level service bekerja efektif dalam menyaring ribuan data historis sesuai parameter yang dibutuhkan untuk kebutuhan audit dan pelaporan.

10. Modul Log System

    Modul Log System dirancang khusus untuk mengelola data log simulasi. Fungsi utamanya adalah melakukan impor massal dari file CSV dan menyajikannya dalam daftar yang dapat dicari. Pengujian ini memastikan parser CSV dapat menangani berbagai format header secara fleksibel (fuzzy header match).

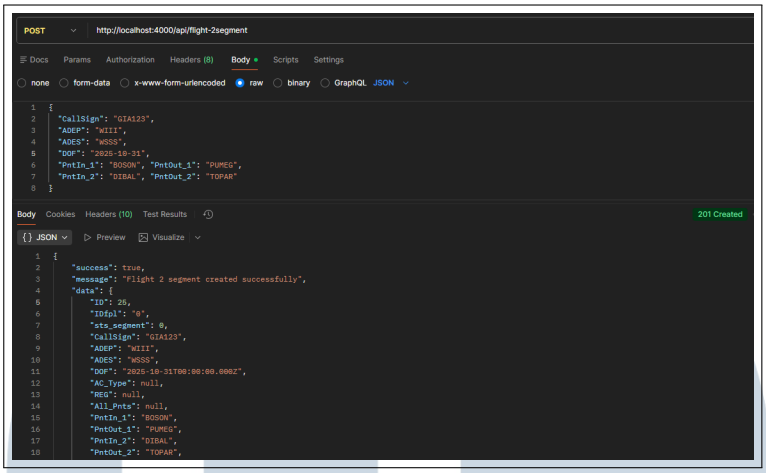


Gambar 3.20. Hasil Pengujian Endpoint Modul Log System Menggunakan Postman

Keberhasilan proses unggah log demo membuktikan bahwa sistem mampu melakukan normalisasi data dari sumber eksternal (CSV) ke skema demo_log_system. Hal ini memungkinkan tim operasional melakukan simulasi analisis data historis untuk keperluan pelatihan atau pengujian rute.

11. Modul Flight 2-Segment

    Menguji struktur data kompleks yang mencakup dua segmen rute dalam satu nomor penerbangan. Pengujian ini memastikan sistem mampu memetakan payload JSON yang lebih luas tanpa terjadi korupsi data.



Gambar 3.21. Hasil Pengujian Endpoint Modul Flight 2-Segment Menggunakan Postman

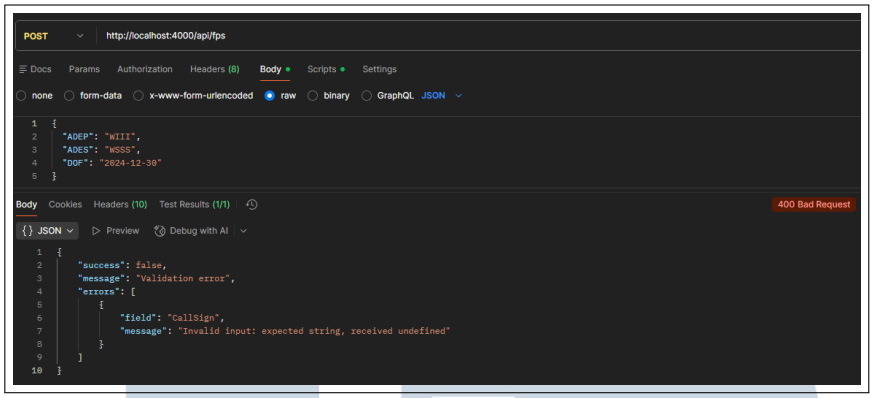
Keberhasilan penyimpanan data 2-segment membuktikan fleksibilitas database HAADES dalam menangani rute penerbangan yang melintasi wilayah FIR Indonesia lebih dari satu kali dalam satu perjalanan.

B Pengujian Validasi

Pengujian validasi sangat penting untuk memastikan bahwa aplikasi menolak input yang tidak valid sebelum memproses lebih lanjut. Setiap field yang memiliki aturan validasi diuji dengan berbagai input yang tidak valid untuk memastikan validasi bekerja dengan benar.

1. Field Wajib

   Pengujian ini memverifikasi bahwa ketika field wajib tidak dikirim, server mengembalikan kode status 400 dengan daftar field yang bermasalah dan pesan kesalahan yang informatif. Struktur error response yang konsisten memudahkan client untuk menampilkan pesan kesalahan kepada pengguna.

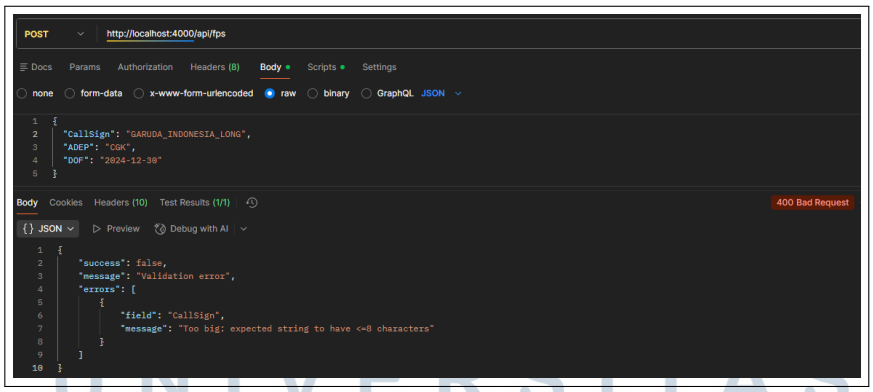


Gambar 3.22. Hasil Pengujian Validasi Field Wajib Menggunakan Postman

Sistem memberikan respons yang sangat spesifik melalui Error Middleware. Pesan ”Validation error” disertai detail field yang kurang membuktikan bahwa Type Safety terjaga, sehingga data cacat tidak akan pernah masuk ke database.

2. Format Data Tidak Valid

   Pengujian dengan CallSign yang terlalu panjang (lebih dari 8 karakter), ADEP yang tidak tepat 4 karakter, dan format tanggal yang tidak valid memastikan bahwa validasi Zod menangkap semua kesalahan format sebelum data diproses oleh Service Layer.

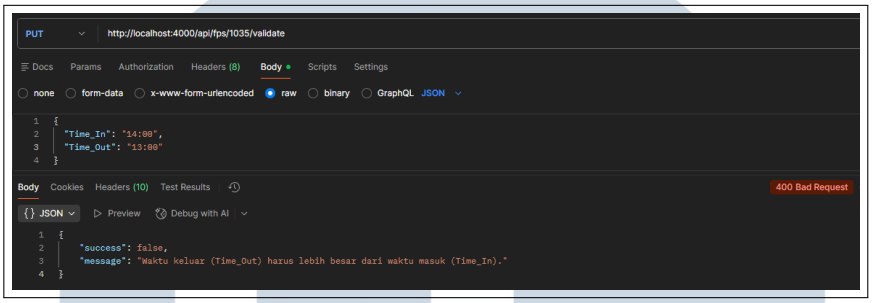


Gambar 3.23. Hasil Pengujian Validasi Format Data Tidak Valid Menggunakan Postman

Kegagalan ini membuktikan bahwa sistem mampu melakukan filter otomatis terhadap kesalahan input manusia (Typo) yang tidak sesuai dengan konvensi penerbangan dunia.

3. Validasi Business Rules

Pengujian validasi aturan bisnis memverifikasi bahwa Service Layer menolak data yang tidak memenuhi aturan bisnis seperti waypoint yang tidak ada di database atau Time_Out yang lebih kecil dari Time_In.



Gambar 3.24. Hasil Pengujian Validasi Aturan Bisnis pada Service Layer Menggunakan Postman

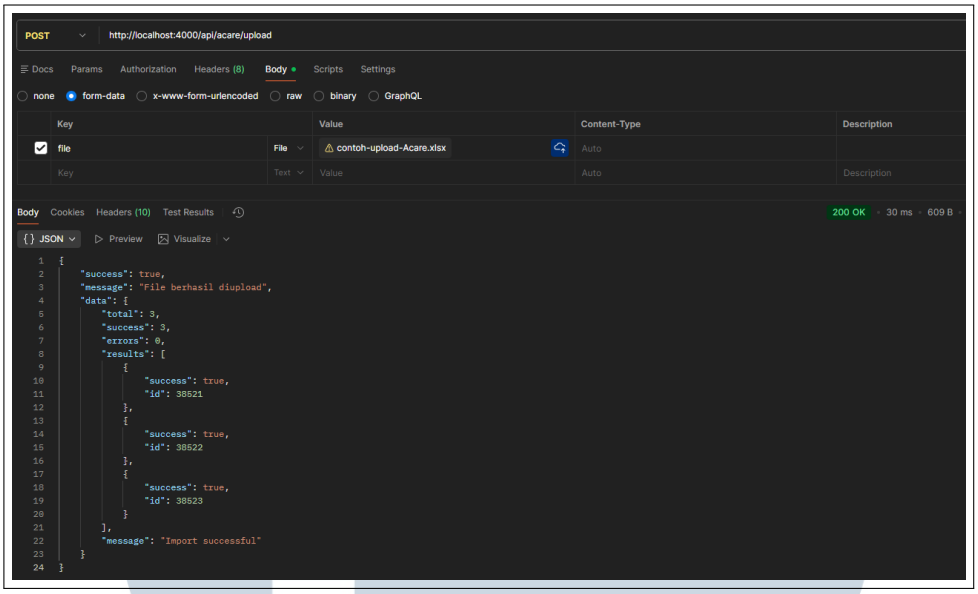
Respons 400 dengan pesan khusus ”Waktu keluar harus lebih besar...” menunjukkan bahwa sistem memiliki logika untuk menjaga kualitas data tagihan, mencegah terjadinya anomali data.

C Pengujian File Upload

Pengujian file upload dilakukan untuk memverifikasi kemampuan sistem dalam menangani unggahan dokumen eksternal, khususnya file Excel (.xlsx) pada modul A-CARE. Pengujian ini mencakup proses penanganan multipart/form-data, ekstraksi data, validasi struktur kolom, serta pemeriksaan setiap baris data sebelum disimpan ke dalam basis data. Fokus pengujian diarahkan pada pembatasan tipe dan ukuran file untuk menjamin keamanan dan performa sistem, serta pada kemampuan sistem dalam menghasilkan laporan kesalahan yang informatif apabila data tidak memenuhi kriteria yang ditetapkan.

1. Upload File Excel Valid

   Skenario ini memverifikasi bahwa sistem mampu memproses file binary dalam format Excel (.xlsx) dengan struktur kolom yang benar (ADEP, ADES, DOF, AC_Type, dll). Keberhasilan pengujian ini memastikan bahwa data dalam jumlah besar dapat diimpor ke tabel data_acare secara otomatis.

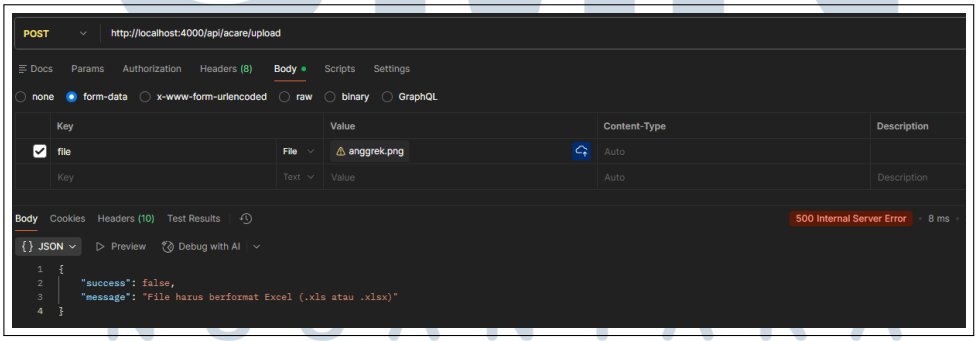


Gambar 3.25. Hasil Pengujian Upload File Excel Valid Menggunakan Postman

Respons sukses (200 OK) dengan jumlah data inserted yang sesuai membuktikan bahwa Parser Excel di tingkat middleware berfungsi dengan benar. Hal ini menghilangkan risiko kesalahan human error yang biasa terjadi pada proses input data manual.

2. Upload File dengan Tipe Tidak Valid

Skenario keamanan ini memastikan sistem menolak file dengan tipe yang tidak diizinkan (misal: .pdf atau .png). Filter pada middleware Multer harus mampu mendeteksi MIME type file sebelum diproses lebih lanjut untuk mencegah risiko serangan siber.



Gambar 3.26. Hasil Pengujian Keamanan Upload File dengan Tipe Tidak Valid

Penolakan sistem terhadap file non-Excel dengan pesan ”Tipe file tidak valid”

memvalidasi lapis keamanan sistem dalam menyaring integritas dokumen yang diunggah ke *server storage*.

3. Upload File dengan Data Sebagian Valid

   Skenario ini menguji ketangguhan sistem saat menghadapi file Excel yang berisi campuran data benar dan data yang melanggar aturan (misal: kode ADEP hanya 2 karakter). Sistem diharapkan dapat memisahkan data yang lolos validasi dan data yang gagal.

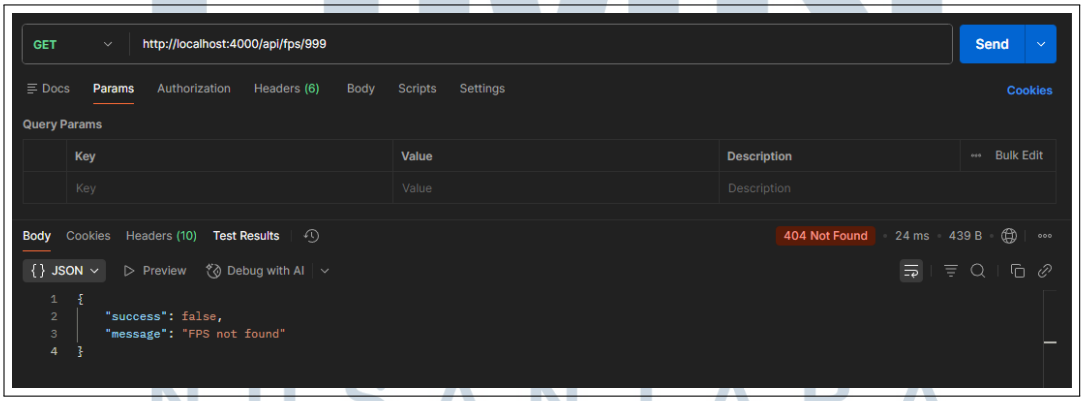
   Keberhasilan sistem melaporkan baris spesifik yang gagal (misal: baris ke-2 gagal karena field ADEP) memberikan pengalaman pengguna yang baik, di mana staf administrasi dapat segera memperbaiki data yang salah tanpa harus mengulang proses unggah seluruh file.

D Pengujian Error Handling

Pengujian error handling memastikan bahwa aplikasi menangani berbagai situasi kesalahan dengan anggun dan memberikan respons yang informatif.

1. Resource Not Found (404)

   Pengujian ini memverifikasi bahwa ketika client meminta resource dengan ID yang tidak ada dalam database, API mengembalikan kode status 404 Not Found dengan pesan error yang informatif. Scenario ini umum terjadi ketika user mengakses link yang sudah tidak valid atau mencoba mengakses data yang sudah dihapus.



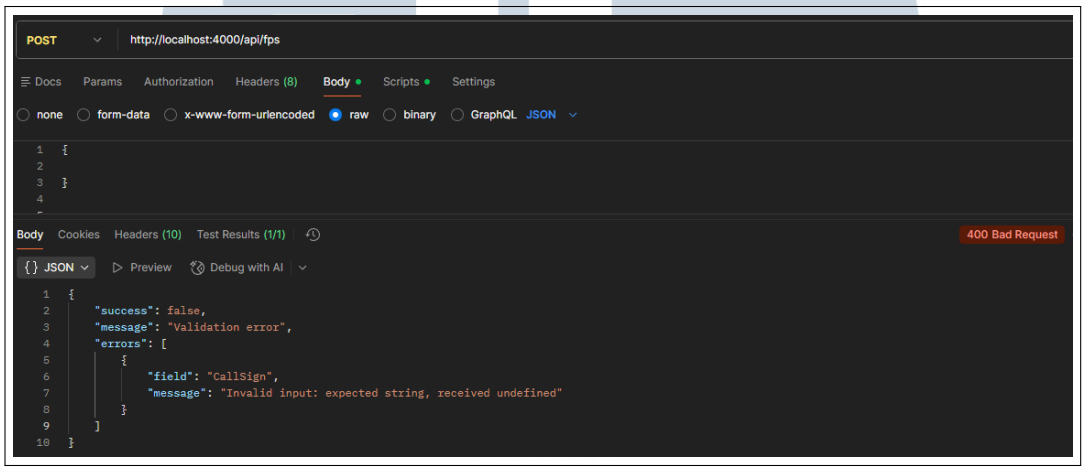
Gambar 3.27. pengujian untuk resource not found

Respons 404 dengan pesan ”FPS not Found” membuktikan bahwa Error Middleware aplikasi mampu memetakan kegagalan query database menjadi

respons yang bermakna bagi client, bukan sekadar error teknis yang membingungkan.

2. Validation Error (400)

   Pengujian ini memverifikasi bahwa API menolak data yang tidak memenuhi schema validasi Zod dan mengembalikan detail error untuk setiap field yang bermasalah. Response harus mencakup array errors yang berisi informasi field dan message untuk setiap validation failure.

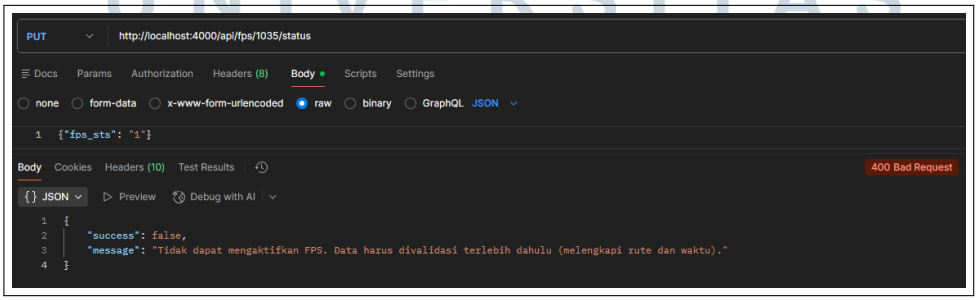


Gambar 3.28. Hasil Pengujian Validation Error (400) pada API Menggunakan Postman

Status 400 Bad Request dengan array errors menunjukkan bahwa sistem secara proaktif mencegah data sampah masuk ke database. Hal ini menjamin integritas data pada tabel-tabel inti seperti all_fpl.

3. Business Logic Error (400)

   Pengujian ini memverifikasi bahwa Service Layer menolak request yang tidak memenuhi business rules meskipun schema validasi sudah passed.

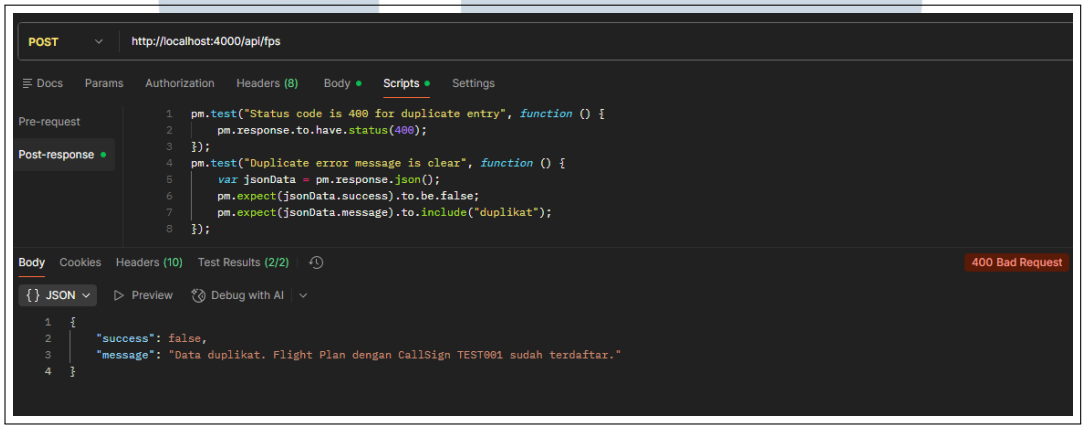


Gambar 3.29. Hasil Pengujian Penolakan Business Rules pada Service Layer

Meskipun format JSON benar, server harus menolak dengan pesan ”FPS belum divalidasi”. Ini memverifikasi bahwa sistem memiliki logika proteksi berlapis yang memastikan urutan operasional (Work Order) ditaati oleh seluruh staf pengguna.

4. Duplicate Entry (400)

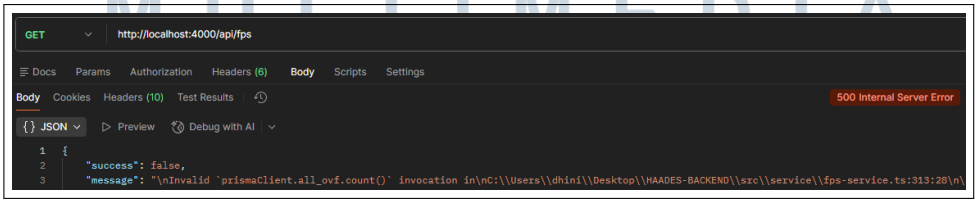
   Pengujian ini memverifikasi bahwa API menolak data yang tidak memenuhi schema validasi Zod dan mengembalikan detail error untuk setiap field yang bermasalah. Response harus mencakup array errors yang berisi informasi field dan message untuk setiap validation failure.



Gambar 3.30. Hasil Pengujian Duplicate Entry (400) pada API Menggunakan Postman

5. Server Error (500)

   Skenario ini menguji pertahanan sistem ketika terjadi kesalahan yang tidak terduga pada server (misal: koneksi database terputus). Sistem harus tetap memberikan respons formal tanpa membocorkan detail teknis yang sensitif. Pengujian untuk memverifikasi bahwa unhandled errors ditangkap oleh error middleware dan dikembalikan sebagai generic 500 error tanpa expose detail internal.



Gambar 3.31. Hasil Pengujian Server Error (500) pada API Menggunakan Postman

3.4 Kendala dan Solusi yang Ditemukan

Selama proses pengembangan dan implementasi Backend HAADES, terdapat beberapa kendala teknis yang muncul dan langkah penyelesaian yang dilakukan sebagai berikut:

(a) Terjadi kendala pada skema database hasil introspeksi, di mana beberapa tabel legacy tidak memiliki Primary Key tunggal yang didefinisikan secara eksplisit. Hal ini menyebabkan penggunaan metode findUnique pada Prisma Client memicu kegagalan sistem karena kriteria keunikan tidak terpenuhi.

(b) Terdapat kebutuhan logika bisnis yang kompleks yang melampaui validasi tipe data dasar, seperti verifikasi keberadaan koordinat titik (waypoint) di database referensi serta validasi temporal (memastikan waktu keluar harus setelah waktu masuk).

Melalui kendala tersebut, Solusi yang dilakukan sebagai berikut:

(a) Untuk mengatasi keterbatasan skema tanpa Primary Key eksplisit, dilakukan peralihan metode dari findUnique menjadi findFirst dengan kriteria pencarian yang diperketat. Pendekatan ini memastikan stabilitas sistem tanpa harus mengubah struktur database utama.

(b) Diimplementasikan lapisan validasi transaksional pada Service Layer yang mengombinasikan Zod untuk validasi skema dasar dan custom business logic untuk validasi lintas tabel (lintas referensi data point) serta pengecekan logika temporal guna menjamin akurasi data tagihan.