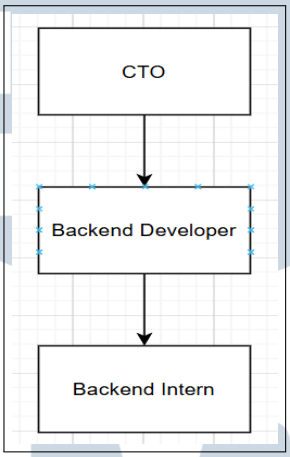
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Selama kegiatan magang, posisi yang dijalani adalah sebagai *Backend Developer Intern* dibawah divisi IT Development. Penugasan dilakukan di bawah arahan langsung dari bapak Daffa sebagai *Chief Technology Officer*, serta berkoordinasi dengan anggota tim lainnya seperti developer Templating Service dan tim *frontend*. Seluruh aktivitas magang difokuskan pada pengembangan layanan *backend*, khususnya dalam membangun Content Service.



Gambar 3.1. Kedudukan tim backend

Koordinasi dilakukan secara rutin melalui rapat mingguan menggunakan platform Google Meet, yang diikuti oleh seluruh anggota tim untuk menyampaikan progres, kendala, serta rencana kerja ke depan. Pelaporan tugas harian dilakukan melalui server Discord sebagai bagian dari dokumentasi dan monitoring pekerjaan secara harian. Komunikasi tambahan dilakukan melalui pesan pribadi kepada leader apabila terdapat kendala teknis atau memerlukan arahan lebih lanjut, baik melalui chat WhatsApp pribadi maupun diskusi dalam grup.

Dalam pelaksanaan magang ini, diskusi aktif juga dilakukan dengan sesama backend developer, khususnya terkait integrasi layanan penyimpanan *template* HTML pada service yang sedang dikembangkan. Selain itu, koordinasi dengan tim *frontend* juga dilakukan untuk memastikan integrasi antara *backend* dan *frontend* berjalan dengan baik dan sesuai kebutuhan sistem.

3.2 Tugas yang Dilakukan

Selama menjalani kegiatan magang, tanggung jawab diberikan dalam proses pengembangan sistem backend untuk layanan *E-Invitation*. Fokus utama pengembangan diarahkan pada pembangunan layanan Content Service yang berperan dalam pengelolaan aset digital dan *template* undangan. Seluruh proses dilakukan dengan menggunakan *framework* Java Spring Boot serta integrasi dengan layanan pihak ketiga seperti Google Drive API. Adapun tugas-tugas yang dilakukan antara lain sebagai berikut:

1. Mengelola penyimpanan data aset digital (gambar dan video) dari berbagai event ke dalam database PostgreSQL.
2. Mengembangkan fitur untuk menyimpan dan mengambil file template HTML undangan dari berbagai jenis event.
3. Melakukan integrasi sistem dengan Google Drive API untuk memungkinkan penyimpanan dan pengambilan file eksternal menggunakan link.
4. Membuat dan mengelola REST API untuk komunikasi antar komponen dalam sistem.
5. Melakukan uji coba (testing) terhadap endpoint-endpoint API untuk memastikan fungsionalitas berjalan dengan baik.

3.3 Uraian Pelaksanaan Magang

Selama masa kerja magang yang berlangsung dari tanggal 3 Maret 2025 hingga Mei 2025, penugasan difokuskan pada peran sebagai *backend developer*. Kegiatan utama yang dilakukan adalah membuat sistem CRUD untuk penyimpanan file mp3 ataupun mp4 sebagai aset undangan, dan membuat sistem CRUD untuk penyimpanan template undangan html pada *local database*. Selain menyimpan pada *local database*, sistem ini juga memanfaatkan GoogleDrive API sebagai metode penyimpanan eksternal. Pelaksanaan kerja magang dirinci dalam tabel di bawah ini:

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1.

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

Minggu Ke -	Pekerjaan yang dilakukan
1	Setup environment, Cloning repository, dan Setup database
2	Membuat api untuk CRUD File/content ke local database dan google-drive , fixing beberapa logic dan mengintegrasi dengan google-drive.
3	Melakukan riset mengenai API Spotify dan Youtube sebagai opsi buat streaming music/video untuk keperluan e-invitation.
4	Melakukan riset mengenai cara menguhungkan content service dengan template service, dan membuat API untuk mengelolah template HTML dalam local database.
5	Mengembangkan API dan logic service untuk mengolah template HTML dengan integrasi Google-Drive API, membuat JDBC repository, dan sedikit bug fixing.
6	Bug fixing untuk logic pengelolahan template berbasis Google-drive dan riset mengenai tehnik caching.
7	Update logic content service untuk menghubungkan API CRUD file ke GoogleDrive
8 - 9	Re-Develop logic Google Drive Service dan melakukan testing terhadap endpoint API.
10	Re-Develop model dan controller dan setup database untuk CRUD Template HTML
11	Update JDBC Content repository dan develop logic ContentService untuk menyimpan Template HTML pada local database
12	Membuat logic GoogleDrive service untuk menghubungkan API CRUD Template HTML ke GoogleDrive
13 - 14	Melakukan testing terhadap endpoint API dan berdiskusi dengan front-end untuk proses integrasi
15	Menambahkan logic auto make folder pada content service setiap ada event baru
16	Fixing auto make folder logic
17	Merapikan variabel penting untuk menjadi hidden dalam .env dan membuat config dan config loader nya

Selama pelaksanaan kegiatan magang, terdapat serangkaian aktivitas teknis yang terbagi ke dalam tiga belas minggu. Aktivitas tersebut dirancang secara bertahap untuk membangun, mengembangkan, serta mengintegrasikan sistem *backend* berupa layanan CRUD API yang mendukung pengelolaan file digital dan template HTML pada website *E-Invitation*. Seluruh proses pengembangan dilakukan menggunakan *framework* Java Spring Boot, dengan PostgreSQL sebagai sistem basis data, serta integrasi penyimpanan eksternal melalui Google Drive API.

Pada minggu pertama dilakukan proses *setup environment* untuk mempersiapkan lingkungan pengembangan proyek. Kegiatan dimulai dengan *cloning repository* dari sistem kontrol versi, lalu dilanjutkan dengan setup database sebagai dasar untuk pengembangan fitur-fitur berikutnya. Tahapan ini menjadi fondasi awal agar proyek dapat berjalan dengan baik dalam *database lokal*.

Minggu kedua difokuskan pada pembuatan fungsi CRUD untuk file atau konten yang tersimpan di database lokal dan Google Drive. Beberapa logika tambahan dikembangkan untuk menangani proses upload file, serta dilakukan integrasi awal dengan Google Drive API. Selain itu, sejumlah bagian kode yang masih bermasalah diperbaiki agar proses pengelolahan file berjalan lebih stabil.

Pada minggu ketiga dilakukan riset mengenai integrasi API eksternal seperti Spotify dan YouTube untuk mendukung fitur streaming musik atau video sebagai pelengkap dalam sistem undangan digital. Riset ini mencakup analisis cara kerja API dan cara mengintegrasikannya. Hasil riset digunakan sebagai referensi untuk kemungkinan integrasi pada fase selanjutnya. Akan tetapi, setelah berdiskusi dengan atasan, akhirnya diputuskan hanya menggunakan Google-Drive sebagai penyimpanan eksternal.

Minggu keempat diisi dengan riset tambahan terkait koneksi antara content service dan template service. Tujuan riset ini adalah untuk menemukan metode terbaik dalam mengelola template HTML dari *local database*. Selain itu, dilakukan juga pengujian awal terhadap struktur penyimpanan dan pemrosesan file template HTML. Pada minggu kelima dikembangkan API dan *logic* dari service untuk mengelola template HTML. Integrasi dengan Google Drive API dilakukan agar template dapat disimpan langsung ke *cloud storage* Google Drive. Selain itu, dibuat juga struktur JDBC *repository* serta dilakukan perbaikan kecil pada beberapa logika program.

Minggu keenam digunakan untuk *bug fixing* terhadap logic pengelolaan template yang terhubung ke Google Drive. Selain perbaikan, juga dilakukan riset teknik *caching* untuk meningkatkan efisiensi pengambilan data. Tujuannya

adalah untuk menghindari pengambilan file berulang dari Google Drive yang menyebabkan keterlambatan sistem. Setelah berdiskusi dengan template service dan atasan, ditetapkan bahwa caching hanya di terapkan oleh template service ketika mengakses *endpoint* API dari content service.

Minggu ketujuh difokuskan pada pembaruan logika content service agar dapat terintegrasi penuh dengan Google Drive API. *Endpoint* CRUD dioptimalkan untuk mendukung pengelolaan file yang lebih efisien. Proses ini juga mencakup penyesuaian struktur data yang dikembalikan oleh layanan Google Drive. Pada minggu kedelapan hingga kesembilan dilakukan redevelop terhadap logic Google Drive Service agar lebih mudah diuji dan memastikan fitur bejalan dengan baik. Setelah itu, dilakukan pengujian intensif terhadap *endpoint* API yang telah diperbarui.

Minggu kesepuluh digunakan untuk pengembangan ulang *model* dan *controller*, serta setup ulang database untuk mendukung operasi CRUD terhadap Template HTML. Database disesuaikan agar dapat menyimpan informasi dan bisa meyimpan path file HTML dan juga menyesuaikan agar event_id pada template sama dengan event_id pada database untuk menyimpan file. Penyesuaian ini ditujukan untuk meningkatkan keterbacaan dan efisiensi akses data.

Minggu kesebelas difokuskan pada pembaruan repository JDBC untuk content dan pengembangan logic ContentService. Fungsi penyimpanan Template HTML pada *local database* ditingkatkan agar lebih andal. Semua proses dilakukan dengan memastikan konsistensi data antar komponen aplikasi.

Pada minggu kedua belas dikembangkan ulang logic Google Drive service agar dapat menangani koneksi ke API CRUD Template HTML. Layanan ini memungkinkan pengelolaan template secara langsung dari dan ke Google Drive. Logic ini juga menerapkan fitur pagination dimana hanya mengambil template untuk nilai yang sudah ditentukan untuk mengurangi beban load. Minggu ketiga belas dan keempat belas digunakan untuk melakukan pengujian menyeluruh terhadap seluruh endpoint API. Selain itu, dilakukan diskusi teknis dengan tim *frontend* terkait proses integrasi antarmuka pengguna. Beberapa penyesuaian dilakukan berdasarkan *feedback* hasil pengujian dan kebutuhan dari sisi *client*.

Minggu kelima belas dilakukan penambahan logic otomatisasi pembuatan folder pada content service berdasarkan nama event. Fitur ini bertujuan untuk mengelola file agar tersimpan secara terstruktur dan mudah diakses. Implementasi logic ini juga menghindari tumpang tindih penyimpanan file antar event.

Pada minggu keenam belas dilakukan perbaikan terhadap logic *auto-make*

folder yang sebelumnya mengalami bug. Fungsi pengecekan eksistensi folder disempurnakan agar tidak menghasilkan duplikasi. Proses ini memastikan bahwa sistem dapat membuat struktur direktori yang benar dan efisien.

Minggu ketujuh belas difokuskan pada perapian variabel penting untuk dimasukkan ke dalam file konfigurasi .env. Pengaturan ini bertujuan untuk menjaga keamanan informasi sensitif seperti folder id dan credential lainnya. Selain itu, dilakukan pembaruan terhadap konfigurasi sistem agar lebih mudah dikembangkan.

3.3.1 Proses Perancangan

Selama masa magang, pekerjaan dilaksanakan sebagai *Backend Developer* yang berkontribusi dalam pengembangan Content Service pada proyek *E-Invitation*. Proyek ini bertujuan untuk menyediakan solusi digital undangan dengan harga yang lebih terjangkau namun tetap mempertahankan kualitas yang kompetitif dengan produk-produk serupa di pasar pernikahan dan event lainnya. Platform ini menawarkan sistem pengelolaan aset digital dan template HTML undangan yang fleksibel dan dapat disesuaikan oleh pengguna, sehingga membutuhkan sistem *backend* yang stabil, efisien, dan terintegrasi.

A Analisis Requirements untuk Admin

1. Manajemen File Aset. Admin dapat mengunggah, memperbarui, melihat, dan menghapus file aset berupa gambar,video, dan musik yang digunakan dalam undangan digital. File yang diunggah akan secara otomatis disimpan di Google Drive dan metadata-nya tercatat di dalam database PostgreSQL. Admin juga dapat melihat link Google Drive dari setiap file yang tersimpan.
2. Manajemen Template HTML. Admin memiliki kemampuan untuk mengelola kumpulan template HTML yang digunakan sebagai desain undangan. Admin dapat:
   (a) Menambahkan template baru (*upload* file HTML),
   (b) Melihat daftar template beserta *preview*-nya,
3. Klasifikasi Berdasarkan Jenis Event. Admin dapat mengelompokkan file dan template berdasarkan jenis event, seperti pernikahan, ulang tahun,dan sebagainya. Setiap ada event baru, maka sistem akan otomatis membuat

folder baru pada Google drive dan mengisi asset-asset yang diperlukan dalam folder tersebut.

B Perangkat Penunjang

Sebelum menjelaskan aktivitas kerja magang secara menyeluruh, pembahasan akan dimulai terlebih dahulu dengan membahas mengenai arsitektur sistem dan perangkat penunjang (baik perangkat keras maupun lunak) yang digunakan selama proses pengembangan sistem. Dalam proyek magang ini, mahasiswa tergabung dalam tim *backend* yang mengembangkan layanan Content Service pada sistem undangan digital *E-Invitation* DIY. Layanan ini bertugas mengelola data aset dan HTML template undangan yang seluruhnya terhubung dan terintegrasi dengan layanan penyimpanan eksternal, yaitu Google Drive, melalui pemanfaatan Google Drive API.

Sistem *backend* ini dibangun menggunakan arsitektur *microservice*, yang memungkinkan tiap layanan memiliki tanggung jawab dan pengelolaan tersendiri, sehingga lebih mudah dalam hal perawatan dan pengembangan berkelanjutan. Salah satu layanan utama yang dikembangkan adalah CRUD API untuk pengelolaan file (gambar, video, dan template HTML) yang digunakan dalam sistem undangan digital. File yang diunggah oleh pengguna disimpan di Google Drive, sedangkan metadata-nya seperti nama file, link file, dan deskripsi disimpan di dalam database PostgreSQL. Untuk membangun dan mengelola logika *backend* ini digunakan *framework* Java Spring Boot, yang memiliki dukungan kuat terhadap pengembangan RESTful API.

Selama pelaksanaan kerja magang, digunakan laptop pribadi ASUS ROG Strix G15 yang dilengkapi dengan prosesor AMD Ryzen 9 6900HX, GPU NVIDIA GeForce RTX 3070 Ti 8GB, RAM 32GB DDR5, serta SSD NVMe 1TB Gen 4. Spesifikasi ini sangat menunjang kegiatan pengembangan aplikasi, terutama dalam hal multitasking, compiling, testing, dan simulasi API dalam waktu singkat. Layar 1080p dengan refresh rate 300Hz juga memberikan kenyamanan visual saat bekerja dalam durasi yang panjang.

Adapun perangkat lunak yang digunakan selama kegiatan kerja magang beserta fungsinya adalah sebagai berikut:

1. IntelliJ IDEA

   IntelliJ IDEA digunakan sebagai *Integrated Development Environment (IDE)* utama dalam pengembangan backend menggunakan bahasa pemrograman

Java dan framework Spring Boot. IntelliJ menyediakan fitur-fitur seperti *auto-completion, error highlighting*, integrasi langsung dengan Maven/Gradle, dan juga kemampuan untuk menjalankan serta men-debug server lokal secara langsung. Kemudahan dan efisiensi yang ditawarkan oleh IntelliJ membantu mempercepat proses pengembangan API selama magang.

2. PostgreSQL.

   PostgreSQL merupakan sistem manajemen basis data relasional yang digunakan sebagai media penyimpanan metadata file. Data yang disimpan mencakup nama file, jenis event, link Google Drive, dan informasi tambahan lainnya. PostgreSQL dipilih karena memiliki kinerja yang andal, kemampuan untuk menangani query kompleks, serta dukungan terhadap koneksi dari Spring Boot melalui JDBC ataupun JPA *(Java Persistence API)*.

3. DBeaver

   DBeaver digunakan sebagai alat bantu dalam mengelola database PostgreSQL secara visual. Dengan antarmuka grafis yang mudah digunakan, DBeaver memudahkan proses *monitoring* tabel, melakukan *query* SQL, serta memeriksa hasil *input* dan *output* dari sistem secara langsung. Tool ini sangat membantu dalam fase pengujian dan debugging selama integrasi backend dengan database.

4. Postman

   Postman merupakan perangkat lunak yang digunakan untuk menguji setiap *endpoint* API yang dikembangkan. Dengan Postman, penulis dapat melakukan simulasi permintaan HTTP (GET, POST, PUT, DELETE) dan memverifikasi respon dari server. Penggunaan Postman juga mempermudah dokumentasi API serta validasi logika dari CRUD file dan integrasi dengan Google Drive.

5. Google Drive dan Google Drive API

   Google Drive digunakan sebagai media penyimpanan utama untuk file asset yang diunggah oleh pengguna. File tidak hanya disimpan secara langsung dalam server lokal, tetapi juga ditransfer ke Google Drive dan dikelola melalui Google Drive API. API ini memungkinkan sistem *backend* untuk mengunggah, membaca, memperbarui, dan menghapus file secara otomatis

dari folder tertentu, serta menghasilkan tautan yang dapat diakses oleh sistem *frontend*.

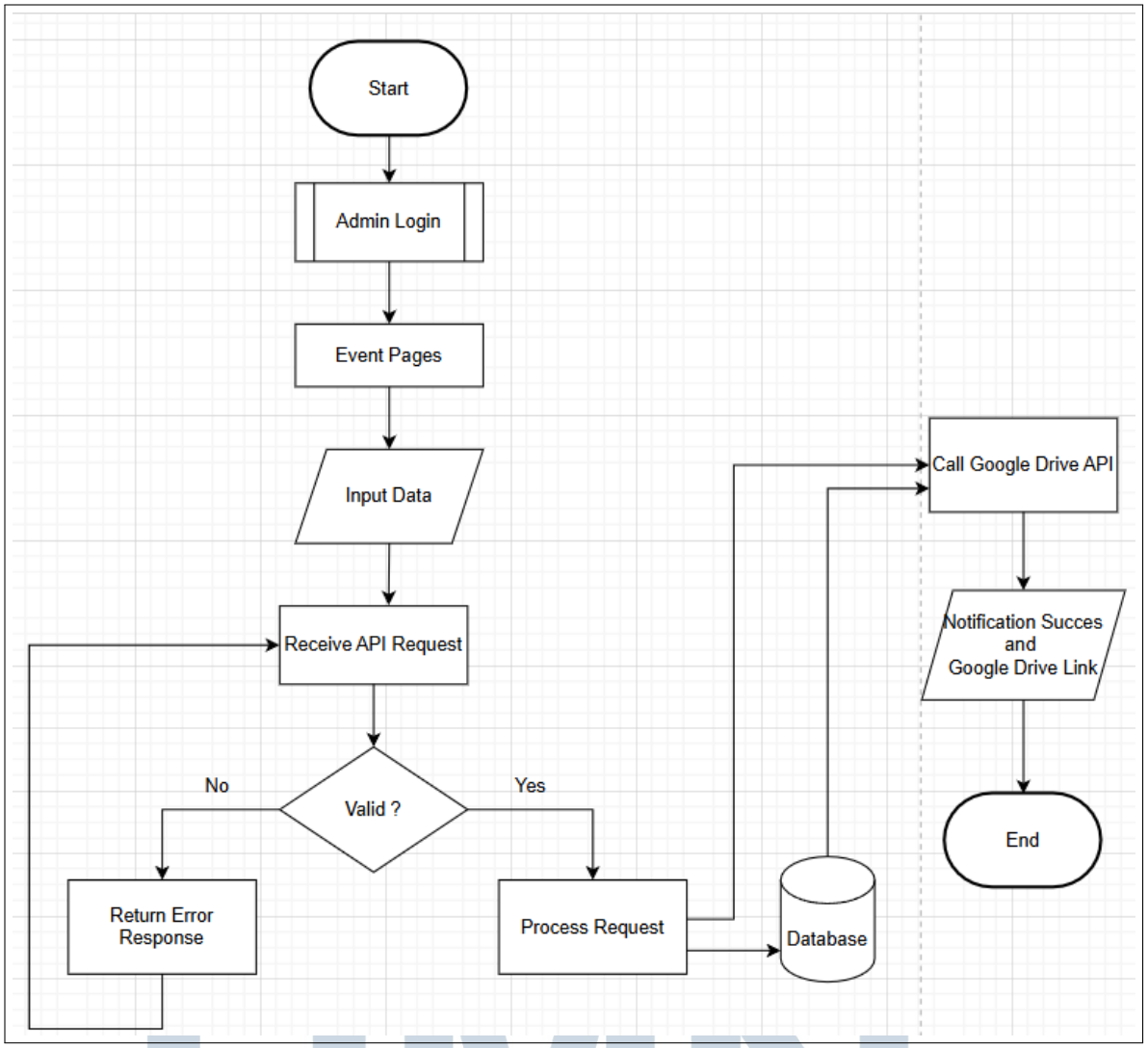
C Flowchart

C.1 Flowchart Content Service

Untuk memahami bagaimana sistem bekerja dalam menangani permintaan dari sisi pengguna hingga penyimpanan data, maka diperlukan pemahaman yang jelas mengenai alur kerja service yang dibangun. Sistem ini menggunakan pendekatan arsitektur berbasis layanan *(service-based architecture)*, di mana setiap operasi CRUD *(Create, Read, Update, Delete)* baik untuk file maupun template HTML ditangani melalui lapisan service yang saling terintegrasi dengan *database* lokal serta layanan penyimpanan eksternal, yaitu Google Drive.

Setiap permintaan *(request)* yang dikirimkan melalui *endpoint* API akan diteruskan ke *controller*, lalu diproses oleh service yang sesuai. Service ini bertanggung jawab untuk menyimpan data ke dalam database PostgreSQL melalui JDBC *Repository* dan juga melakukan integrasi dengan Google Drive, seperti mengunggah file atau membuat folder secara otomatis bila belum tersedia. Proses ini memastikan bahwa data tidak hanya tersimpan secara lokal, tetapi juga dapat diakses secara *cloud* melalui akun Google Drive yang telah terintegrasi.

Agar terdapat gambaran yang lebih jelas mengenai bagaimana sistem memproses setiap permintaan, berikut disajikan *flowchart* alur kerja service untuk fitur CRUD file dan template HTML:



Gambar 3.2. Flowchart Alur Kerja Content Service

Gambar 3.2 merupakan flowcart alur kerja dari asset management pada Content Service. Flowchart ini menggambarkan alur sistem *backend* dalam menangani permintaan terkait data event dan file yang diunggah oleh admin. Alur dimulai ketika admin melakukan login melalui antarmuka sistem. Setelah berhasil autentikasi, admin dapat mengakses halaman manajemen event dan kemudian mengisi beberapa dan juga mengupload file yang akan diinput kedalam database dan Google Drive. Halaman ini juga memungkinkan berbagai aksi, seperti mengunggah file terkait event, mendapatkan semua file yang terkait dengan event, memperbaharui file, dan menghapus file. Setiap aksi ini dikirim dalam bentuk API *request* ke server.

Pada tahap selanjutnya, server menerima dan memvalidasi permintaan API tersebut. Validasi ini mencakup autentikasi token pengguna, kelengkapan parameter

yang dibutuhkan seperti event_id, dan pemeriksaan struktur data. Jika validasi gagal, sistem akan langsung mengembalikan respons berupa *error message*. Namun jika validasi berhasil, maka permintaan akan diproses lebih lanjut tergantung pada jenis aksinya: POST, GET, PUT, dan DELETE.

Fokus utama dari sistem ini adalah proses manajemen file di Google Drive yang berbasis pada event_id. Ketika permintaan POST dikirim untuk membuat event baru, sistem akan terlebih dahulu memeriksa apakah event tersebut sudah pernah dibuat sebelumnya. Pemeriksaan ini dilakukan melalui *database*, di mana event_id menjadi kunci utama identifikasi. Jika nama event belum ada di *database*, maka sistem akan menggunakan Google Drive API untuk membuat folder baru dengan nama sesuai nama event dari *request* untuk menyimpan aset-aset yang diperlukan untuk event terkait didalam folder tersebut.

Setelah folder berhasil dibuat, folder ini akan digunakan kembali untuk aksi lain seperti upload file, update event, maupun delete event, sehingga sistem tidak perlu membuat folder baru setiap kali ada aksi terhadap event yang sama. Dengan cara ini, sistem menjamin bahwa satu event hanya memiliki satu folder unik di Google Drive, menghindari duplikasi data dan memastikan struktur penyimpanan tetap rapi.Sebaliknya, jika event dengan event_id yang sama sudah ditemukan di database, maka sistem tidak akan membuat folder baru. Sebagai gantinya, ia akan menggunakan folder yang sudah ada berdasarkan ID yang tersimpan. Hal ini sangat krusial dalam efisiensi sistem karena mengurangi interaksi yang tidak perlu dengan Google Drive API.

Pada proses GET, ketika admin mengirimkan permintaan untuk mengambil seluruh konten atau file berdasarkan id dari event, server akan terlebih dahulu memvalidasi autentikasi dan kelengkapan parameter. Jika validasi berhasil, server akan mengecek database untuk memastikan event tersebut ada dan mendapatkan folder Google Drive yang terkait. Selanjutnya, server memanggil Google Drive API untuk mengambil daftar semua file yang tersimpan di folder tersebut. Daftar file beserta *metadata* seperti id file, id event, nama file, tipe file, dan link akses kemudian dikirimkan sebagai respons kepada admin agar dapat melihat seluruh file yang terkait dengan event tersebut.

Pada proses DELETE, ketika admin ingin menghapus file tertentu di dalam folder event, server akan memvalidasi permintaan yang berisi event_id dan identifikasi file yang akan dihapus. Jika valid, server akan menggunakan Google Drive API untuk menghapus file spesifik tersebut dari folder Google Drive yang sesuai. Setelah file berhasil dihapus, server mengirimkan konfirmasi penghapusan

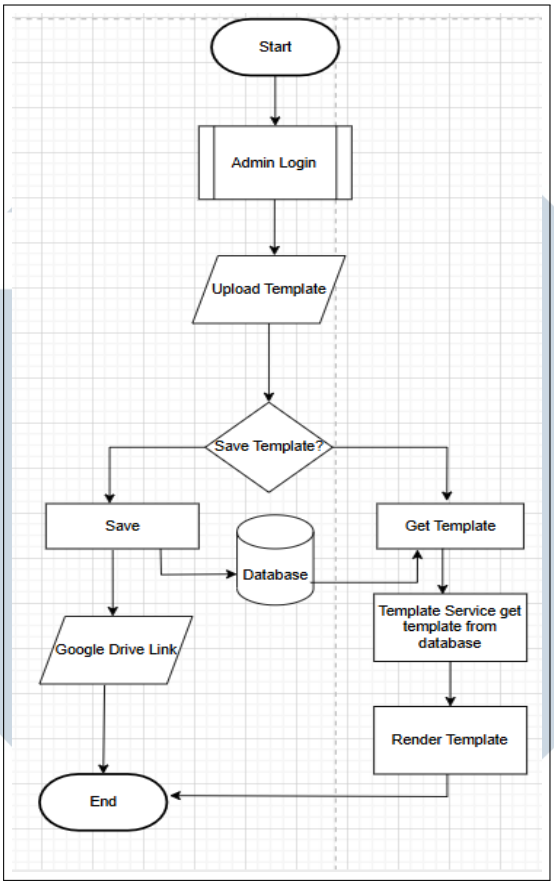
kepada admin. Apabila file tidak ditemukan atau penghapusan gagal, server akan mengembalikan pesan error yang sesuai. Dengan mekanisme ini, sistem mampu mengelola file dalam folder Google Drive yang terstruktur berdasarkan event dengan efisien dan terorganisir.

Untuk proses PUT, admin mengirimkan permintaan pembaruan file dengan menyertakan event_id dan informasi file yang akan diperbarui. Server memvalidasi permintaan tersebut dan memastikan folder Google Drive yang sesuai dengan event tersedia. Setelah itu, server memanfaatkan Google Drive API untuk menggantikan file lama dengan file baru yang dikirimkan oleh admin di dalam folder event. Setelah file berhasil diperbarui, server membentuk metadata file terbaru dan mengirimkan respons konfirmasi keberhasilan pembaruan kepada admin.

Setiap kali file diunggah, file akan langsung ditempatkan ke dalam folder yang sesuai dengan event_id tersebut. Sistem memanggil *endpoint* Google Drive API untuk melakukan unggahan file ke folder yang dituju, menggunakan event_id sebagai referensi. Setelah proses selesai, link atau URL Google Drive folder akan dibentuk dan dikirim kembali sebagai bagian dari respons API. Link ini menjadi output akhir yang dapat digunakan oleh admin atau user untuk mengakses konten dari event tersebut secara langsung.

C.2 Flowchart Pengelolahan template

Dalam pengelolaan event berskala besar, penggunaan template HTML yang konsisten dan mudah disesuaikan menjadi salah satu kebutuhan penting. Untuk itu, sistem manajemen template ini dirancang agar memudahkan Admin dalam menyimpan dan menggunakan kembali template HTML sesuai kebutuhan setiap event. Sistem ini melibatkan beberapa komponen layanan (service) yang saling berinteraksi dengan peran yang jelas. Penjelasan berikut menggambarkan bagaimana alur komunikasi dan pertukaran data terjadi antara Admin, Content Service, dan Templating Service, serta bagaimana setiap bagian bekerja sesuai fungsinya. Dengan alur ini, pengelolaan template menjadi lebih efisien, terstruktur, dan mudah dikembangkan ke depannya



Gambar 3.3. Flowchart Alur Pengelolahan Template Service

Gambar 3.3 merupakan flowcart alur kerja dari template management pada Content Service. *Flowchart* ini menjelaskan alur interaksi antara Admin, Content Service, dan Templating Service dalam sistem manajemen template berbasis HTML untuk keperluan event tertentu. Proses dimulai dari admin, yang memiliki akses untuk menyimpan template HTML tertentu melalui *endpoint* khusus yang disediakan oleh Content Service. Saat admin mengirimkan template HTML untuk sebuah event, request tersebut masuk ke *endpoint* save-template, di mana sistem akan menyimpan HTML tersebut berdasarkan id event. Template ini disimpan secara ganda, yakni di dalam database internal untuk kebutuhan *metadata* dan di Google Drive sebagai bentuk file HTML. Google Drive akan mengembalikan id file, yang kemudian dikonversi menjadi URL link. Link ini dikaitkan dengan id file dan disimpan bersama data template dalam *database*.

Content Service hanya akan bekerja secara aktif dalam satu skenario lain, yaitu ketika Templating Service mengakses *endpoint* get-template-by-id. Dalam skenario ini, Templating Service akan mengirimkan permintaan ke Content Service untuk mengambil HTML template berdasarkan template_id. Setelah menerima

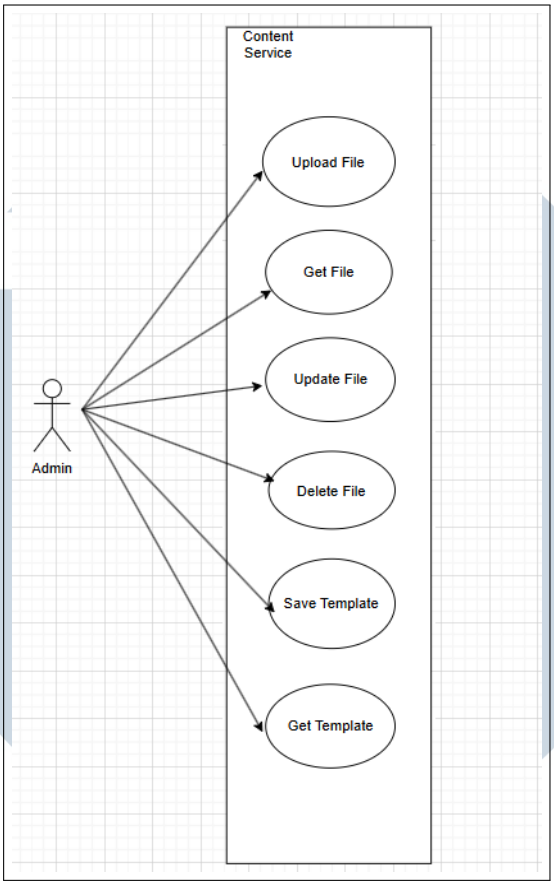
permintaan ini, Content Service akan mengambil HTML template dari database dan juga memastikan file terkait di Google Drive tersedia. Kemudian, HTML template tersebut dikembalikan ke Templating Service. Selanjutnya, Templating Service akan melakukan proses *render* terhadap HTML tersebut menggunakan datanya sendiri atau elemen dinamis lainnya, dan hasil akhirnya dikirim kembali ke admin dalam bentuk template yang sudah di-*render* secara utuh.

Dalam sistem ini, Content Service tidak berjalan secara otomatis, melainkan hanya merespons permintaan yang datang dari endpoint tertentu. *Endpoint* pertama, yaitu save-template, digunakan oleh admin untuk menyimpan HTML template ke dalam sistem berdasarkan event_id. Template ini kemudian disimpan di database serta diunggah ke Google Drive. *Endpoint* kedua adalah GET template-by-id, yang digunakan oleh Templating Service untuk mengambil template HTML mentah berdasarkan ID template yang diminta. Selain itu, terdapat juga endpoint untuk GET all template, untuk mengambil semua template yang sudah dibuat. Permintaan ini kemudian dijawab oleh Content Service dengan mengembalikan isi HTML yang sesuai, yang nantinya akan di-*render* oleh Templating Service. Mekanisme ini menjaga agar Content Service hanya aktif saat diperlukan.

Melalui alur ini, Content Service bertindak sebagai penyimpan dan penyedia data template yang terstruktur dan dapat diakses ulang dan proses rendering sepenuhnya dilakukan oleh Templating Service. *Output* akhir dari alur ini adalah tautan Google Drive yang menyimpan file HTML template, serta respons HTML mentah yang siap untuk di-*render* oleh service lainnya.

D Use Case Diagram

Dalam content service, admin dapat melakukan beberapa hal untuk mengolah file dan template yang dapat dilihat melalui diagram berikut:

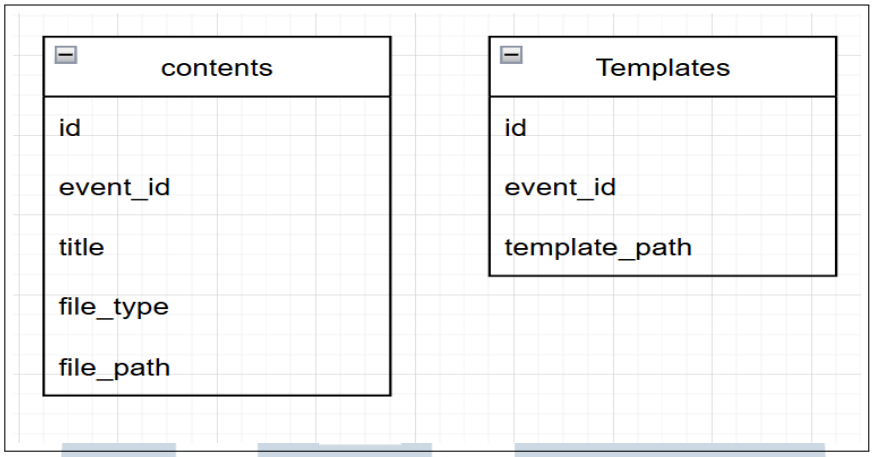


Gambar 3.4. User Diagram

Gambar 3.4 menunjukkan user diagram yang merupakan fitur apa saja yang dapat dilakukan oleh admin dalam Content Service ini.

E Skema Database

Dalam sistem ini, manajemen data yang baik sangat penting karena melibatkan berbagai jenis file dan informasi yang berkaitan dengan event. Oleh karena itu, dibutuhkan struktur basis data yang jelas dan terorganisir agar layanan seperti Content Service dan Template Service dapat saling terintegrasi dengan lancar. Content Service bertanggung jawab untuk menyimpan berbagai file terkait event (seperti gambar atau dokumen), sementara Template Service berfokus pada penyimpanan dan pengambilan template HTML untuk kebutuhan rendering tampilan undangan atau konten visual lainnya.Berikut merupakan *Entity Relationship Diagram* (ERD) untuk sistem manajemen konten dan template:



Gambar 3.5. Skema Database Content Service

Tabel 3.5 merupakan tabel yang dibutuhkan dalam pembuatan Content Service. Tabel contents menyimpan berbagai file yang diunggah dan dikaitkan dengan event tertentu. Setiap baris data dalam tabel ini memiliki id sebagai primary key, event_id sebagai nama event, title sebagai judul atau deskripsi file, file_type untuk mendeskripsikan jenis file (misalnya gambar, dokumen, dll.), serta file_path sebagai jalur akses atau lokasi file tersebut yang biasanya disimpan di Google Drive.

Tabel terakhir adalah templates, yang menyimpan data terkait template HTML yang digunakan untuk keperluan rendering tampilan atau undangan digital. Tabel ini juga memiliki id sebagai primary key, event_id sebagai nama event untuk mendeskripsikan bahwa template tersebut dibuat untuk event tertentu , dan template_path sebagai lokasi atau link file HTML template yang disimpan, umumnya juga dalam bentuk link ke Google Drive.

F Struktur Tabel

F.1 Table: contents

Tabel 3.2. Struktur Tabel *contents*

Nama	Tipe Data	Deskripsi
id	serial	Primary key, tidak boleh NULL.
title	varchar	Menyimpan nama aset dalam basis data, tidak boleh NULL.
event_id	varchar	Menyimpan nama event yang digunakan sebagai nama folder di Google Drive, tidak boleh NULL.
file_type	varchar	Menyimpan tipe file dari aset.
file_path	varchar	Menyimpan link Google Drive dari file aset.

F.2 Table: templates

Tabel 3.3. Struktur Tabel *templates*

Nama	Tipe Data	Deskripsi
id	serial	Primary key, tidak boleh NULL.
event_id	varchar	Mendeskripsikan bahwa template tersebut dibuat untuk event tertentu, tidak boleh NULL.
drive_file_link	varchar	Menyimpan link Google Drive dari file template.

3.4 Hasil dan Implementasi

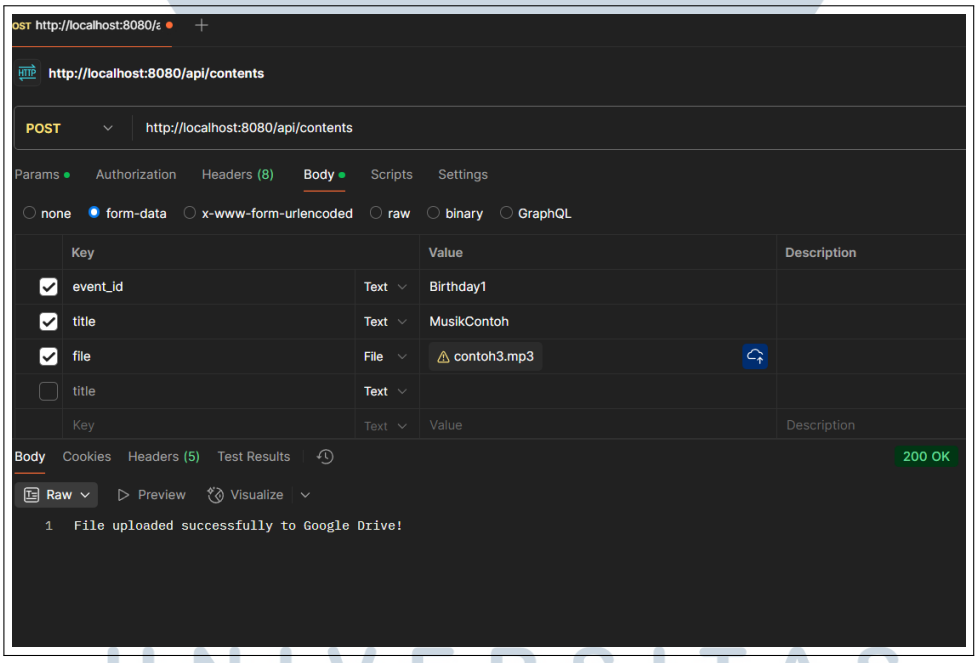
Setelah tahap perancangan dan pengembangan layanan selesai dilakukan, proses selanjutnya adalah mengimplementasikan dan menguji fungsionalitas dari setiap *endpoint* API. Implementasi ini mencakup empat operasi utama yaitu Create (POST), Read (GET), Update (PUT), dan Delete (DELETE), yang dirancang untuk mengelola data konten serta interaksinya dengan penyimpanan lokal maupun layanan Google Drive. Untuk pengelolahan template hanya mencakup metode POST untuk menyimpan template dan GET untuk mengambil semua template. Uji coba dilakukan melalui tools seperti Postman untuk memastikan bahwa seluruh *endpoint* dapat berjalan sesuai dengan kebutuhan sistem dan memberikan hasil yang diinginkan.

3.4.1 Content Service File Management

Content Service untuk file *management* berfungsi untuk menangani request untuk pengelolahan berbagai macam file atau asset yang dibuthkan untuk keperluan undangan, seperti video, musik, atau gambar. Berikut adalah *endpoint-endpoint* untuk mengelolah aset yang ada:

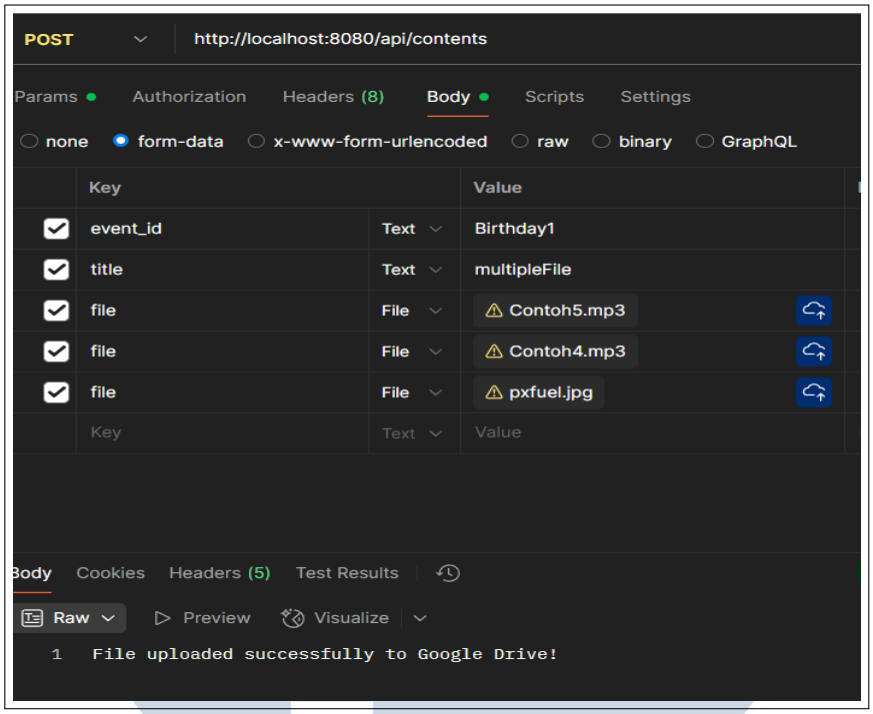
A Post File

Endpoint POST berfungsi untuk mengunggah assset ke dalam database dan juga mengunggahnya ke dalam Google-Drive. *Endpoint* ini juga akan membuat folder untuk menampung aset berdasarkan jenis event yang ada. Jika event yang dimasukkan belum ada sebelumnya, maka endpoint ini akan secara otomatis membuat folder baru sesuai nama event id nya. Berikut cuplikan *endpoint* nya:



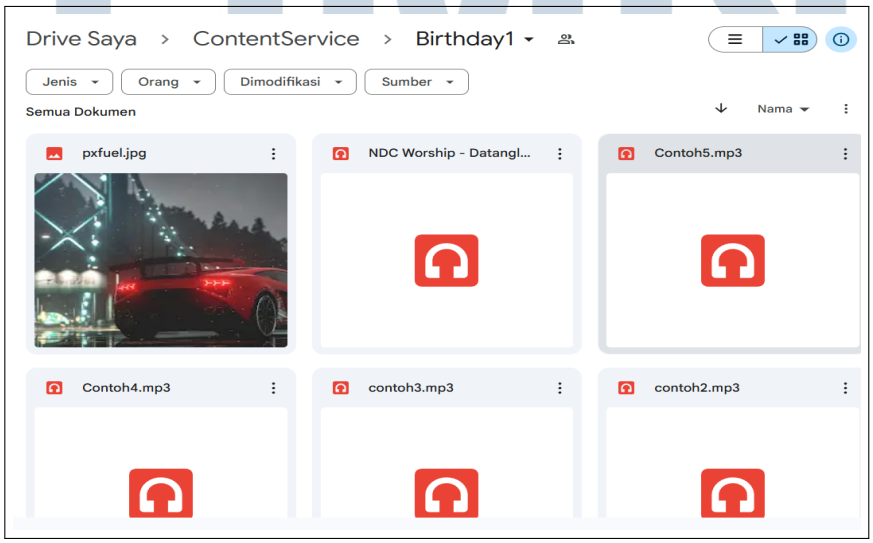
Gambar 3.6. POST api

Gambar 3.6 menunjukkan bahwa endpoint ini berhasil membuat folder dengan nama Birthday1 dan terdapat sebuah file audio yang akan diupload dalam folder tersebut. *Endpoint* ini juga memungkinkan kita untuk mengunggah beberapa file sekaligus agar proses mengunggah file dapat berjalan dengan lebih efisien.



Gambar 3.7. POST api multiple file

Pada gambar 3.7, event_id merupakan nama event yang akan diadakan, misalkan seperti acara nikahan atau ulang tahun. Kemudian ada parameter title yang berfungsi sebagai nama file, dan juga parameter file dengan tipe file yang merupakan file yang akan kita upload. Ketika upload file berhasil maka, akan muncul notifikasi *"File Upload successfully to Google Drive"*. Dan jika kita lihat pada Google-Drive, maka akan muncul folder Birthday1 sesuai nama event yang dituliskan, beserta file-file yang diupload tadi.

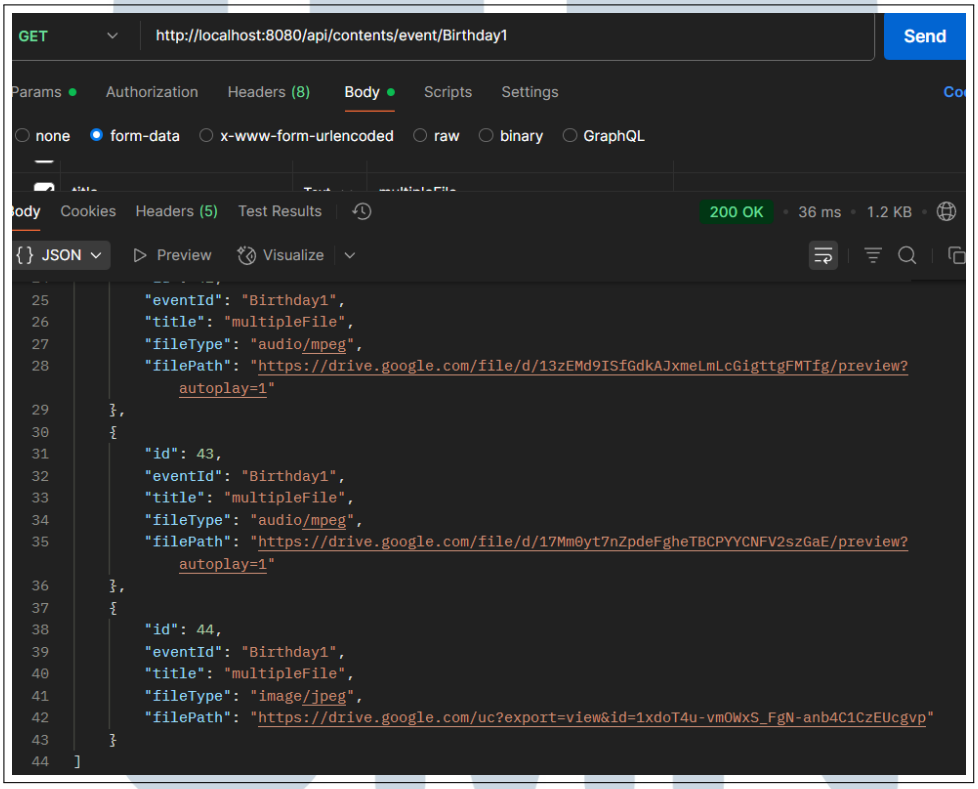


Gambar 3.8. Google Drive Upload Implementation

Gambar 3.8 menunjukkan bahwa file-file yang diupload sebelumnya telah sukses diupload ke dalam folder Google Drive.

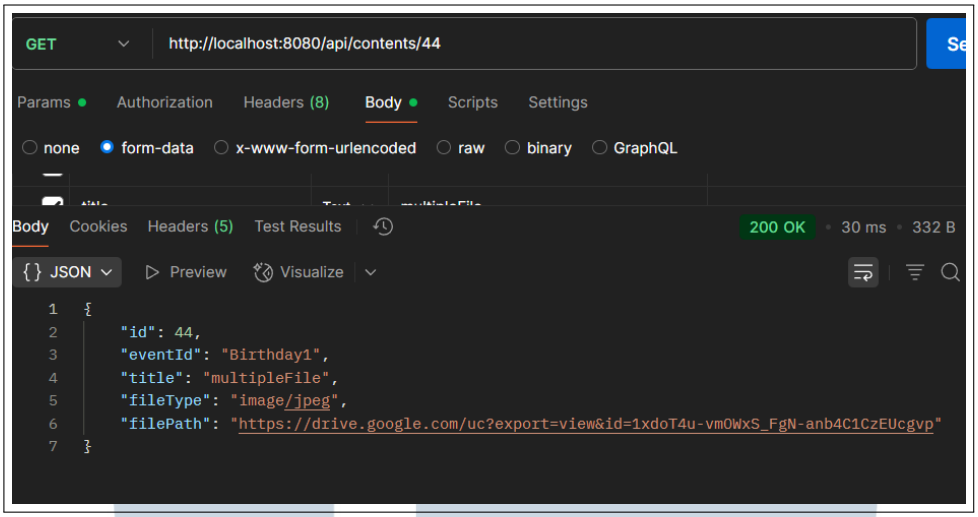
B Get File

Endpoint GET berfungsi untuk mendapatkan atau mengambil semua asset yang sudah kita upload pada endpoint POST, dan kemudian akan memberikan output berupa nama event file tersebut berasal, judul filenya, tipe file, dan juga link google-drive yang mengarah ke file tersebut.



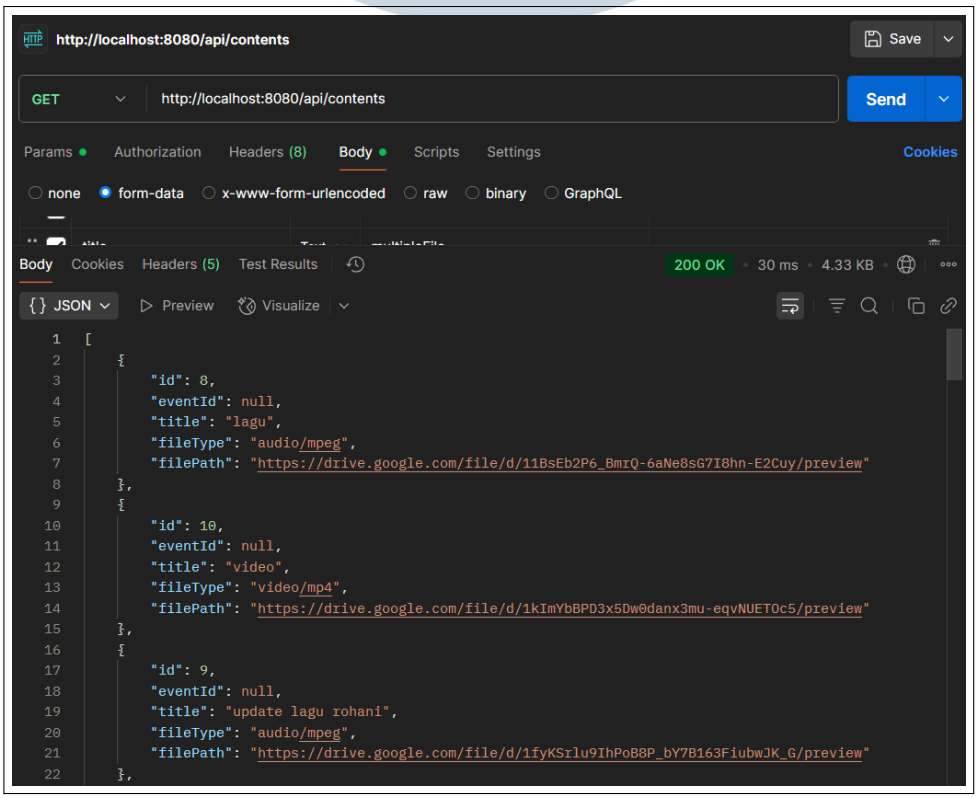
Gambar 3.9. Get Assets by Event

Gambar 3.9 menunjukkan bahwa *endpoint* GET dapat memanggil semua file yang terdapat di dalam folder yang dituju dan juga memberikan link Google Drive dari masing-masing file tersebut. *Endpoint* GET juga dapat melakukan GET by id dimana endpoint ini berfungsi untuk memanggil asset secara spesifik berdasarkan id file nya. Sebagai contoh, berikut merupakan contoh pemanggilan asset dengan id 44.



Gambar 3.10. Get Assets by id

Gambar 3.10 menunjukkan bahwa endpoint GET juga dapat memanggil file berdasarkan id file tersebut. *Endpoint* ini juga dapat memanggil semua assets yang sudah diupload, dari folder manapun untuk ditampilkan. Dapat dilihat pada gambar dibawah, semua asset mulai dari id yang paling kecil hingga paling bawah akan terpanggil.



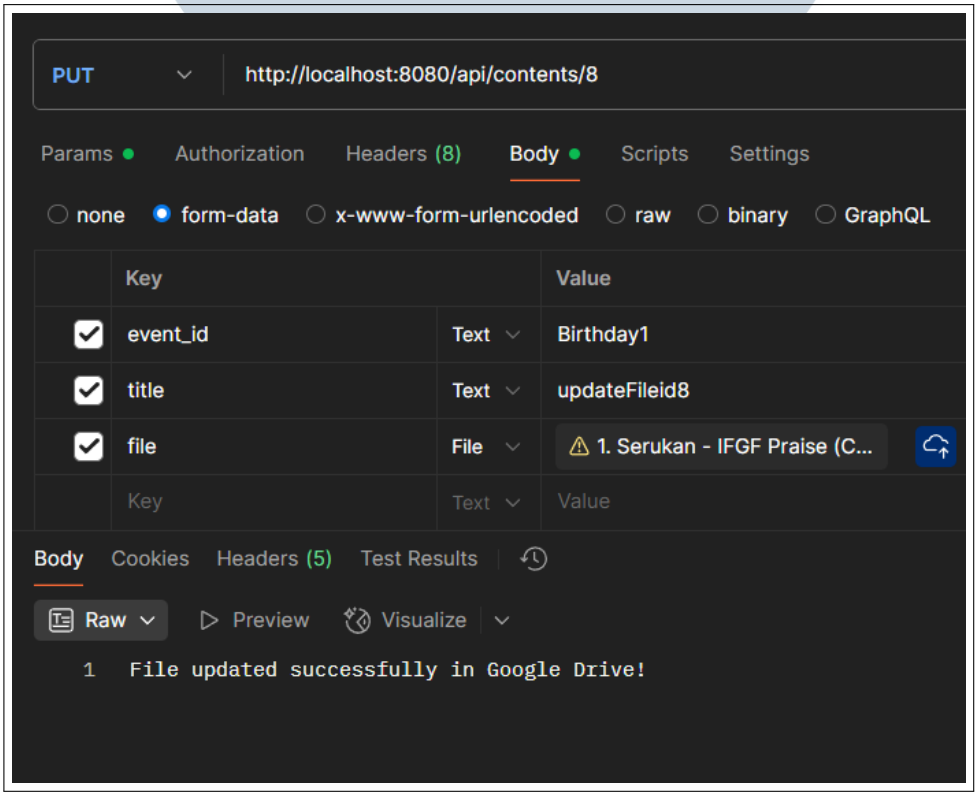
Gambar 3.11. Get All Assets

Gambar 3.11 menunjukkan bahwa endpoint GET juga dapat memanggil

semua aset dari folder manapun untuk ditampilkan, dan memberikan link Google Drive yang mengarah ke file tersebut.

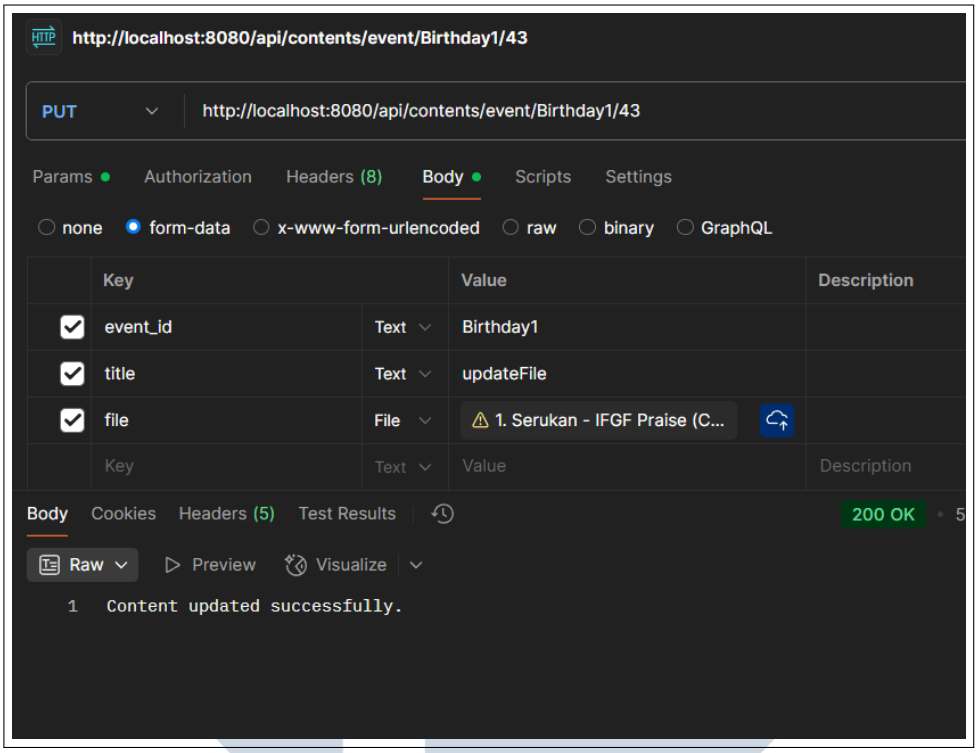
C Update File

Endpoint PUT berfungsi untuk melakukan pembaruan file, seperti perubahan nama, ataupun perubahan file. Endpoint ini dapat melakukan update asset hanya dengan menginput id dari file yang akan diupdate dan juga dapat melakukan update file didalam folder tertentu. Saat melakukan upload file, kita dapat mengupdate nama file dan juga jenis file yang akan diupload ulang. Event_id merupakan lokasi dimana file yang akan diupdate tersebut berada, dan jika nama event_id nya sama seperti sebelumnya, maka endpoint ini tidak akan membuat event dan folder google drive yang baru. Sebagi contoh, berikut akan dilakukan update pada asset dengan id 43.



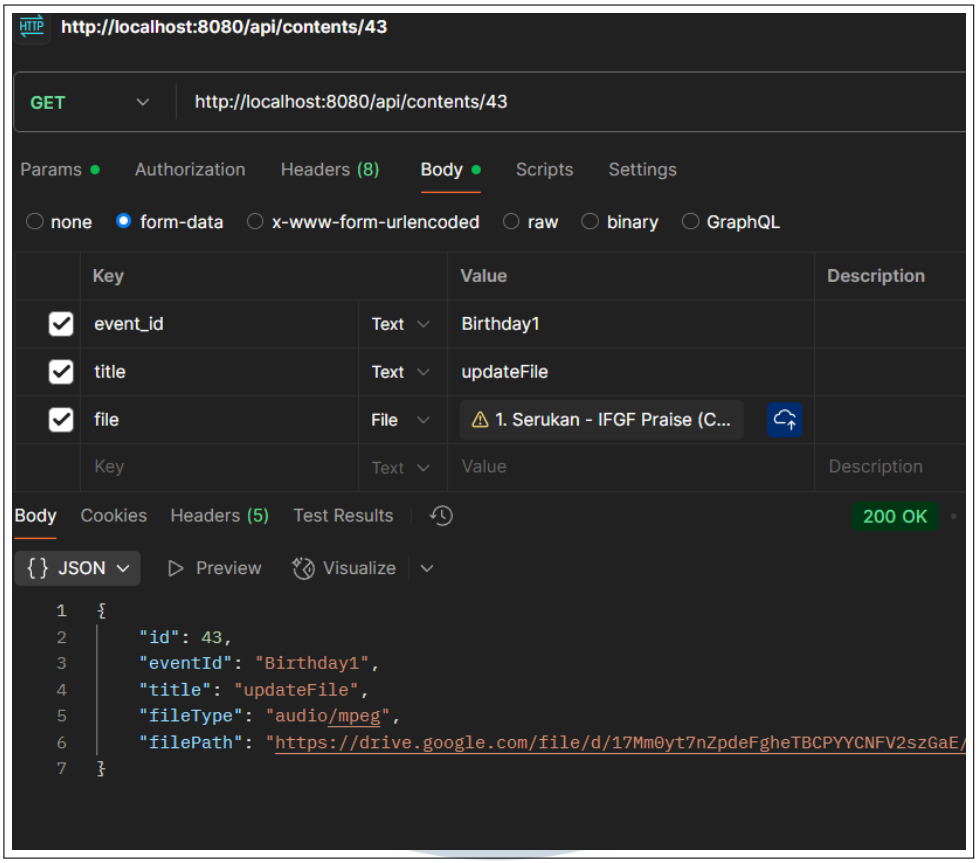
Gambar 3.12. Update by id

Gambar 3.12 menunjukkan bahwa endpoint PUT dapat melakukan update file berdasarkan id file tersebut, dan jika berhasil maka akan memberikan notifikasi bahwa file tersebut sukses diupdate. Selain update by id, terdapat juga endpoint untuk mengupdate file berdasarkan event id.



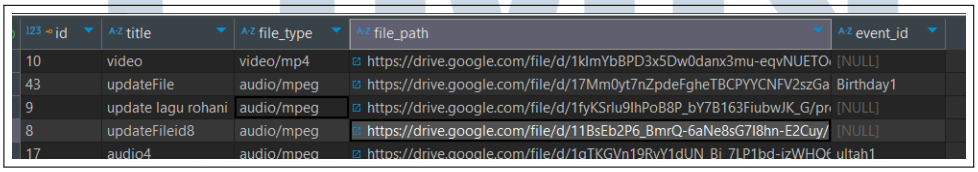
Gambar 3.13. Update by event id

Gambar 3.13 menunjukkan proses update file melalui nama event yang dituju. Setelah proses update, jika kita melakukan GET untuk event Birthday1, maka dapat dilihat bahwa file dengan id 43, sudah diperbaharui, dan perubahan tersebut juga terjadi pada database dan Google-drive. Pada Google drive , nama file mp3 dari asset yang lama adalah contoh 4, akan tetapi setelah diperbaharui, file mp3 tersebut digantikan oleh file dengan nama yang diupdate sebelumnya.



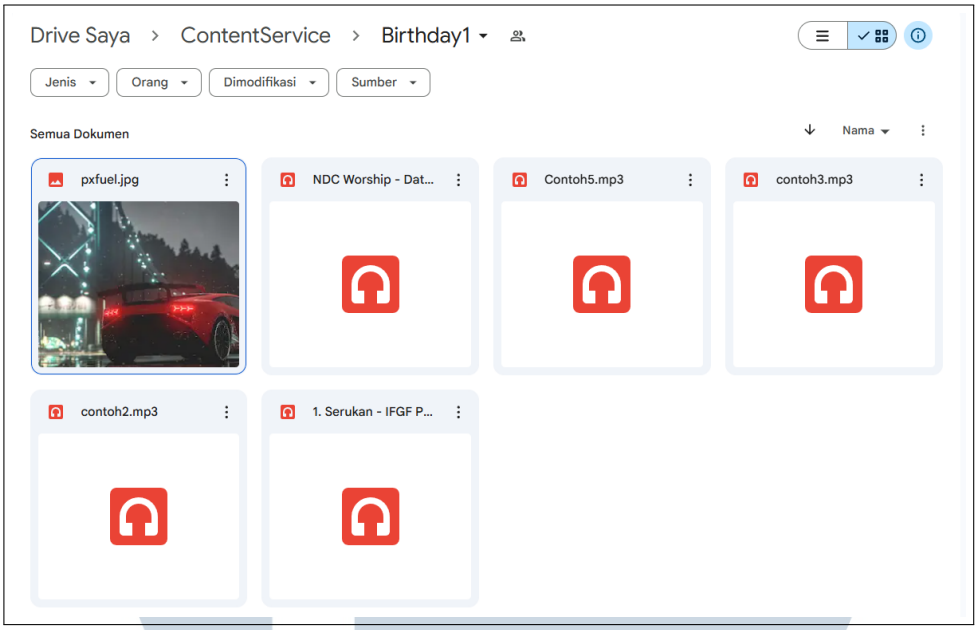
Gambar 3.14. Get Updated assets

Gambar 3.14 menunjukkan bahwa file yang diperbaharui sukses untuk dipanggil kembali dan memberikan link Google Drive yang sudah diperbaharui dari file tersebut. Setelah proses update selesai diproses, maka file di dalam folder google drive dan database akan diperbaharui



Gambar 3.15. Database Updated

Gambar 3.15 menunjukkan database yang telah diperbaharui dengan file yang baru.

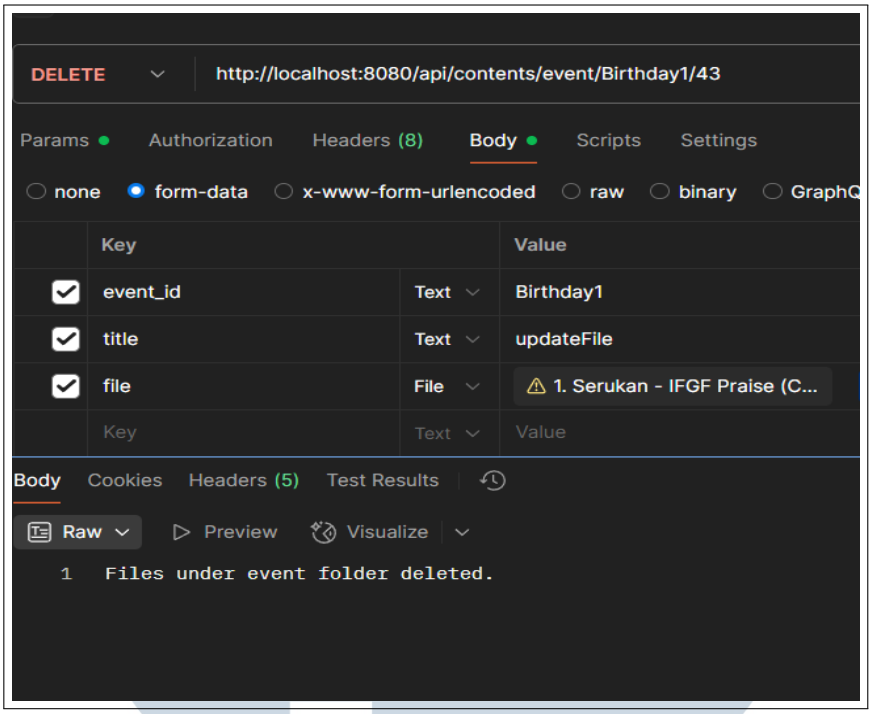


Gambar 3.16. Google Drive Updated

Gambar 3.16 menunjukkan folder Google Drive yang telah diperbaharui dengan file yang baru diupload.

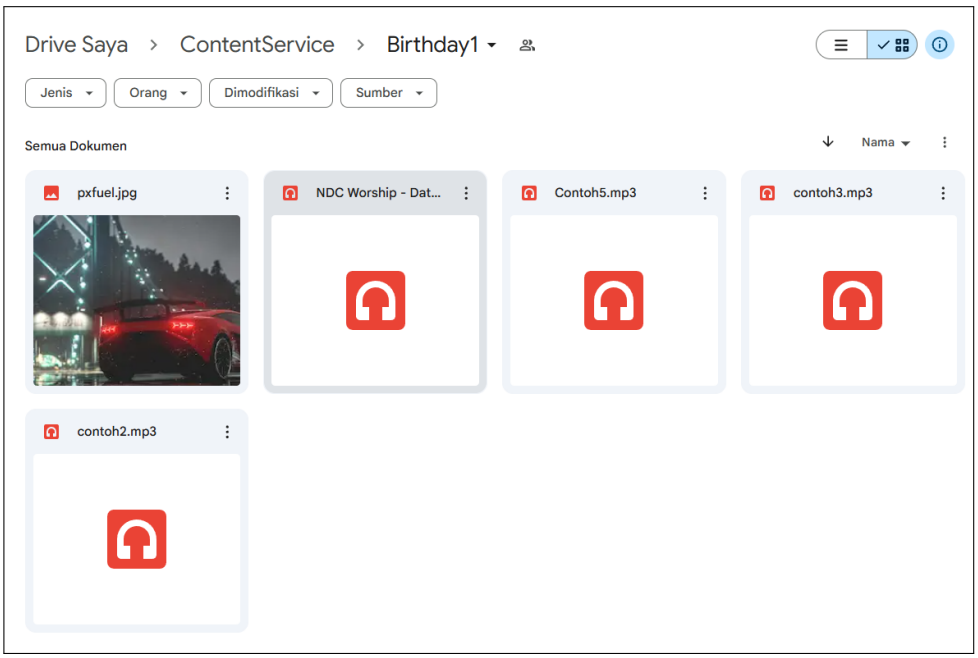
D Delete File

Endpoint DELETE berfungsi untuk menghapus file sesuai dengan id file tersebut. Jika file tersebut dihapus, maka file tersebut akan dihapus juga dari database dan google-drive. Berikut adalah simulasi delete file dengan id 43.



Gambar 3.17. Delete Assets

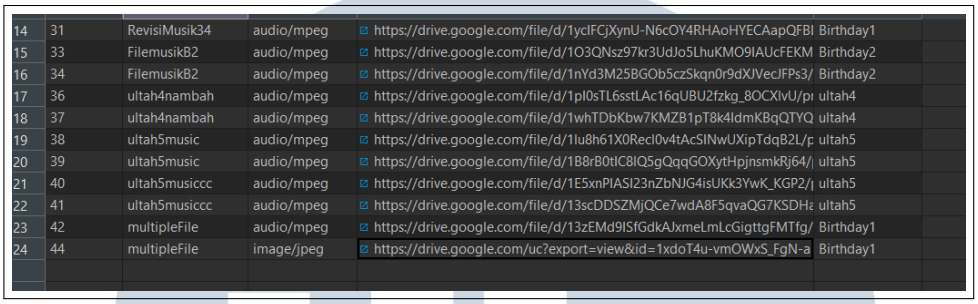
Gambar 3.17 menunjukkan bahwa file dengan id 43 berhasil di hapus. Setelah *endpoint* delete diproses, maka asset yang ada di dalam google drive akan terhapus.



Gambar 3.18. Google Drive Delete Assets

Gambar 3.18 menunjukkan bahwa file dalam folder Google Drive berhasil

dihapus. Selain di google drive, file yang ada di dalam table contents juga akan ikut terhapus.



14	31	RevisiMusik34	audio/mpeg	https://drive.google.com/file/d/1yclFCjXynU-N6cOY4RHAoHYECAapQFBI	Birthday1
15	33	FilemusikB2	audio/mpeg	https://drive.google.com/file/d/1O3QNsz97kr3UdJo5LhuKMO9IAUcFEKM	Birthday2
16	34	FilemusikB2	audio/mpeg	https://drive.google.com/file/d/1nYd3M25BGOb5czSkqn0r9dXJVecJFPs3/	Birthday2
17	36	ultah4nambah	audio/mpeg	https://drive.google.com/file/d/1pI0sTL6sstLAc16qUBU2fzkg_8OCXIvU/p	ultah4
18	37	ultah4nambah	audio/mpeg	https://drive.google.com/file/d/1whTDbKbw7KMZB1pT8k4IdmKBqQTYQ	ultah4
19	38	ultah5music	audio/mpeg	https://drive.google.com/file/d/1lu8h61X0RecI0v4tAcSINwUXipTdqB2L/p	ultah5
20	39	ultah5music	audio/mpeg	https://drive.google.com/file/d/1B8rB0tIC8IQ5gQqqGOXytHpjnsmkRj64/	ultah5
21	40	ultah5musiccc	audio/mpeg	https://drive.google.com/file/d/1E5xnPIASI23nZbNJG4isUKk3YwK_KGP2/	ultah5
22	41	ultah5musiccc	audio/mpeg	https://drive.google.com/file/d/13scDDSZMjQCe7wdA8F5qvaQG7KSDHa	ultah5
23	42	multipleFile	audio/mpeg	https://drive.google.com/file/d/13zEMd9ISfGdkAJxmeLmLcGigttgFMTfg/	Birthday1
24	44	multipleFile	image/jpeg	https://drive.google.com/uc?export=view&id=1xdoT4u-vmOWxS_FgN-a	Birthday1

Gambar 3.19. Database Delete Assets

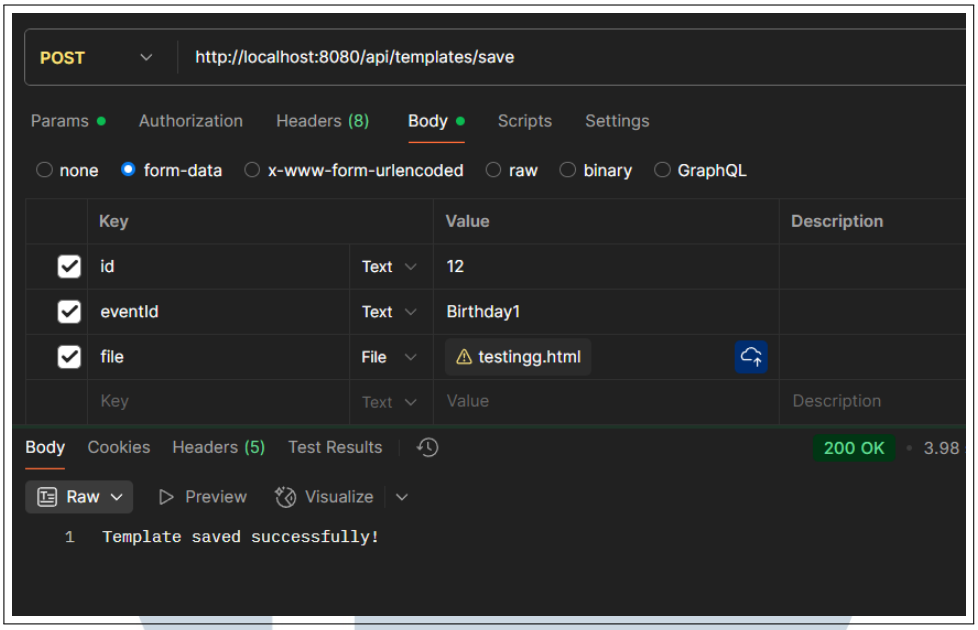
Gambar 3.19 menunjukkan bahwa file dengan id 43 berhasil dihapus dari database.

3.4.2 Content Service Template Management

Service ini berfungsi untuk mengolah template HTML yang akan di render oleh templating service. Ketika templating service mengakses *endpoint* ini, maka *output*-nya akan berupa link google drive yang dibisa digunakan oleh template service

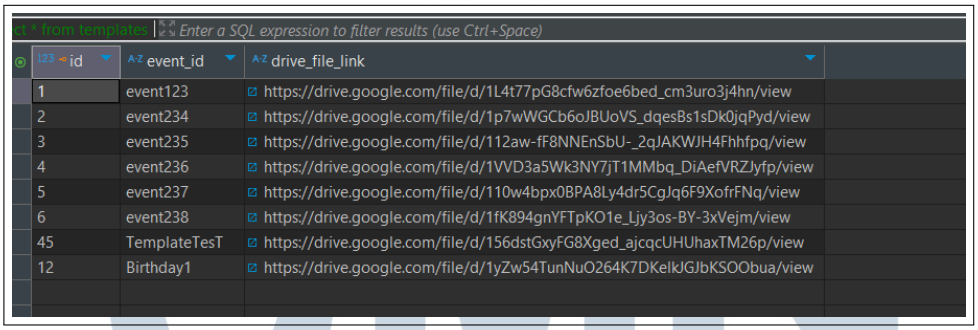
A Save Template

Endpoint Save Template (POST) berfungsi untuk mengunggah template ke dalam database dan juga mengunggahnya ke dalam Google-Drive. Adapun beberapa parameter yang dapat di-*input* seperti id, event_id, dan file html template. Berikut merupakan cuplikan API nya.



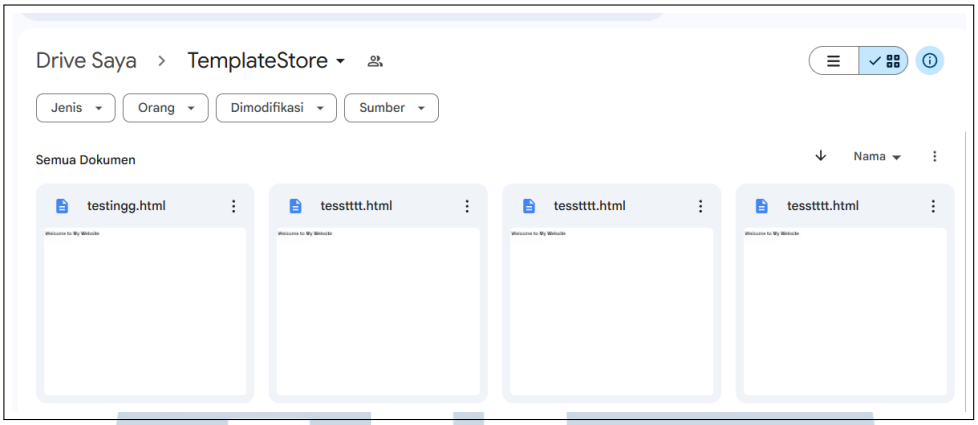
Gambar 3.20. Save Template

Gambar 3.20 menunjukkan bahwa template telah berhasil disimpan ke dalam database dan Google Drive. Setelah *endpoint* save diproses, maka template akan disimpan di dalam database.



Gambar 3.21. Databse Template

Gambar 3.21 menunjukkan bahwa template yang di save berhasil masuk dalam database. Selain database, template html juga akan disimpan di dalam google drive berdasarkan nama event id.

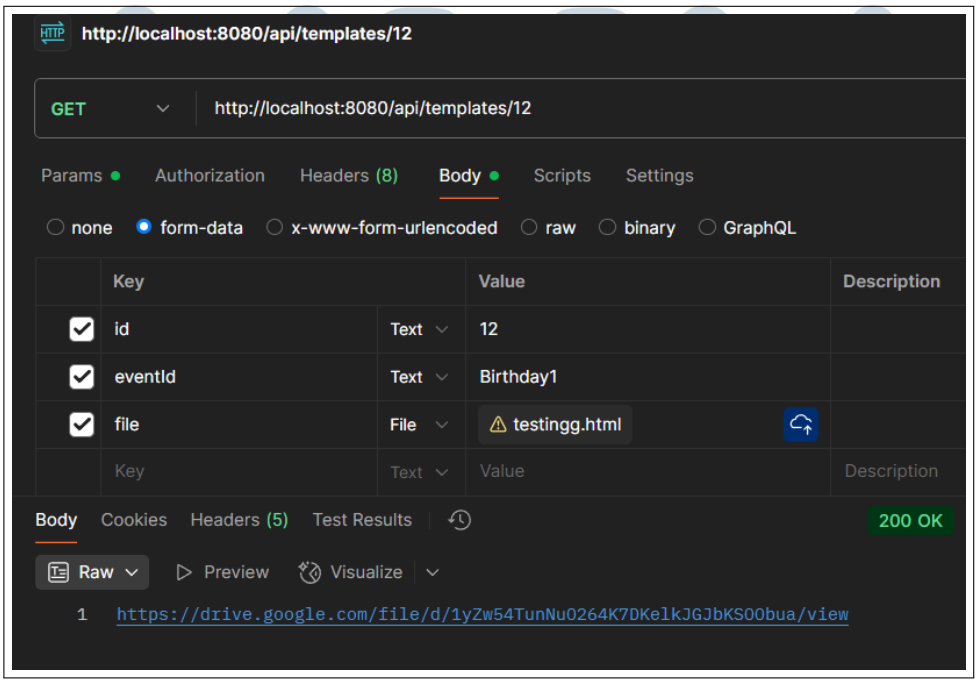


Gambar 3.22. Googledrive Store Template

Gambar 3.22 menunjukkan bahwa template yang di save berhasil masuk dalam Google Drive.

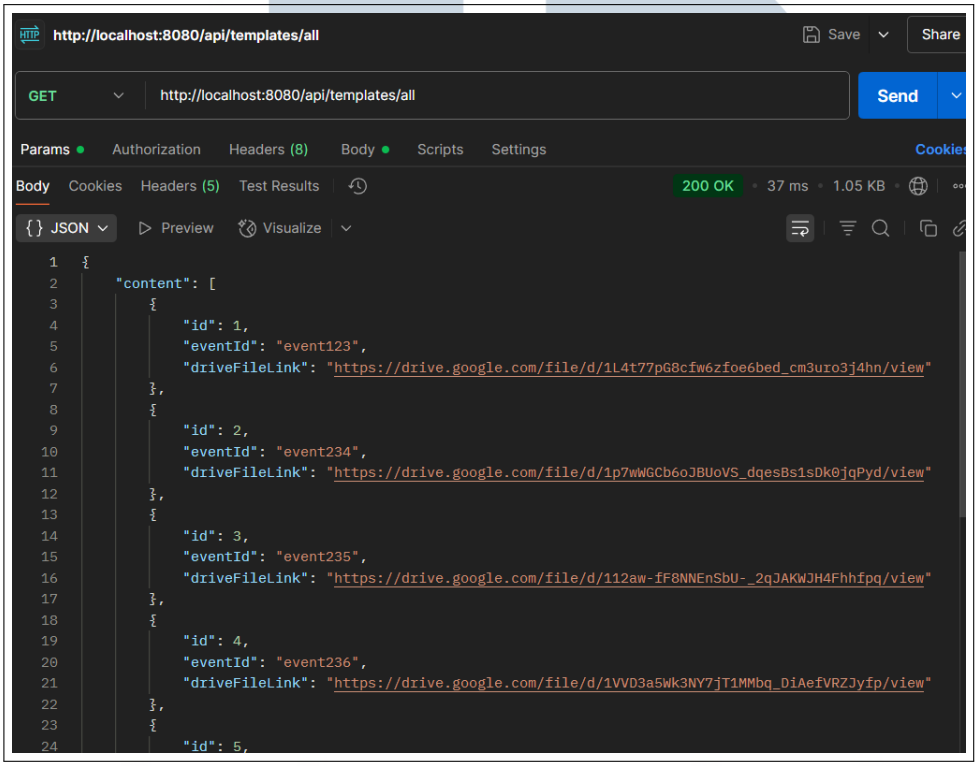
B Get Template

Endpoint ini berfungsi untuk mengambil template berdasarkan id template yang sudah di save dari database. Setelah *endpoint* get selesai diproses, maka akan dihasilkan output yang berupa link google drive dari template yang dipanggil. *Endpoint* ini kemudian akan diakses oleh template service untuk me-*render* template.



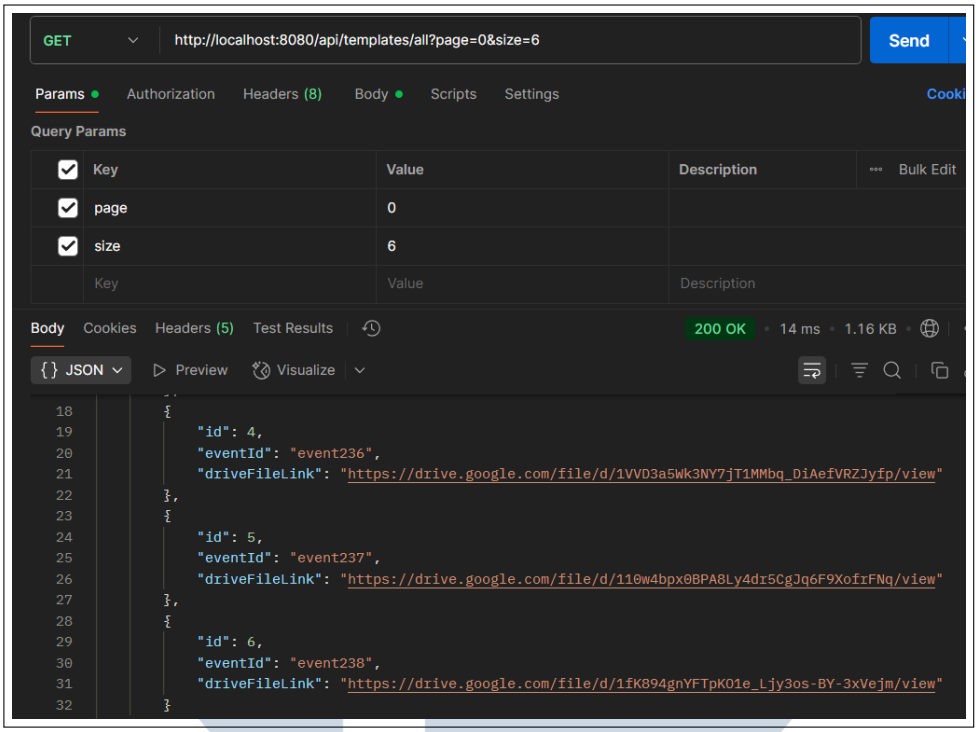
Gambar 3.23. Get Template by id

Gambar 3.23 menunjukkan bahwa endpoint GET dapat mengambil file berdasarkan id nya dan memberikan output berupa link Google Drive dari file tersebut.*Endpoint* ini juga dapat digunakan untuk mengambil semua template yang ada dari database.



Gambar 3.24. Get All Template

Gambar 3.24 menunjukkan bahwa endpoint GET dapat mengambil semua template dari database dan memberikan output berupa link Google Drive dari file tersebut, namun hanya terbatas oleh 5 file untuk meringankan beban kerja website. *Endpoint* ini juga mengaplikasikan fitur pagination untuk membatasi jumlah template yang diambil agar proses tidak terlalu besar.



Gambar 3.25. Template Pagination

Gambar 3.25 menunjukkan fitur pagination yang memungkinkan admin untuk memanggil template berdasarkan jumlah maksimal yang diharapkan.

3.5 Kendala dan Solusi yang Ditemukan

A Kendala yang Ditemukan

1. Pemahaman terhadap *framework* Java Spring Boot yang terbatas. Hal ini membuat mulainya proses pengembangan aplikasi tertunda.
2. Pemahaman terhadap implementasi API yang terbatas. Hal ini membuat integrasi sistem dengan Google Drive menjadi tertunda.
3. Koordinasi antar anggota tim yang belum maksimal pada awal masa magang. Hal ini menyebabkan proses evaluasi dan pelaporan progres menjadi kurang terstruktur.

B Solusi yang Ditemukan

1. Membaca dokumentasi serta video relevan tentang Java Spring Boot, sehingga pengembangan aplikasi bisa dilakukan dengan lancar.

2. Membaca dokumentasi serta video relevan tentang Google Drive API.
3. Diadakannya pertemuan rutin setiap dua minggu melalui WhatsApp atau Discord yang membahas mengenai progres pengembangan.