

## BAB 3 PELAKSANAAN KERJA MAGANG

### 3.1 Kedudukan dan Koordinasi

Dalam masa periode kerja magang di PT Pharos Indonesia sebagai *Staff Backend Engineer (Internship Level)* dibawah supervisi Pak Steven sebagai kepala divisi IT dan *Project Manager* di PT Pharindo (Anak perusahaan PT Pharos) serta diawasi oleh senior *Backend Engineer(Supervisor Backend)* dalam pengerjaan tugas. Dalam proses kerja, Pak. Steven menentukan *flow* pekerjaan dan pembagian tugas untuk divisi *Backend Engineer* dan *Frontend Engineer* lalu senior *Backend Engineer* akan membagikan tugas yang diberikan oleh Pak. Steven kepada anggota - anggota tim per divisi. Para senior akan bertanggung jawab atas proses pengerjaan tugas yang telah dibagikan kepada anggota - anggota.

Untuk koordinasi terdapat beberapa media komunikasi yang digunakan masing - masing dengan kegunaan tersendiri, diantaranya adalah :

- Google Meets : Sebagai sarana media komunikasi yang digunakan saat meeting formal yang diadakan 2 kali seminggu.
- Discord : Sebagai sarana media komunikasi yang digunakan oleh tim untuk mengadakan diskusi atau meeting informal selama WFH.
- Ruang Diskusi : Sebagai tempat yang digunakan tim untuk diskusi secara informal atau individual sat sedang WFO.
- Slack : Sebagai media *webhook* yang memberi notifikasi kepada tim Backend jika Frontend mengalami masalah dengan API yang memberi internal server error.

### 3.2 Tugas yang Dilakukan

Pada proyek CenturyNet Admin, langkah awal yang dilakukan pada saat mulai memasuki proyek adalah memahami struktur *database* melalui *Entity Relational Diagram*. Tidak lupa untuk menyiapkan *environment* dan *tools* yang diperlukan untuk mengerjakan fitur yang ditugaskan, alat - alat yang digunakan adalah :

- Github

- DBeaver
- Visual Studio Code
- Slack
- Firebase
- ThunderClient

Setelah melakukan instalasi dan konfigurasi pada setiap alat - alat yang dibutuhkan untuk workspace yang akan digunakan. Selanjutnya, peserta magang melaksanakan *briefing* untuk mengenal struktur data dan *flow process* yang ada dalam proyek CenturyNet Admin. Briefing ini menjelaskan struktur data menggunakan grafik *Entity Relational Diagram*, tujuannya agar dapat mengerti sifat - sifat data dan relasi antar tabel yang telah berjalan sebelum pelaksanaan magang di dalam proyek.

### 3.3 Uraian Pelaksanaan Magang

Requirement yang diberikan oleh Project Manager kepada *Intern Backend Engineer* adalah sebagai berikut :

1. Membuat API *Get List Of Ads Notification By Query, Filter, Sort, and Pagination* yang berfungsi untuk mengambil data semua iklan berdasarkan query, filter, sort dan pagination.
2. Membuat API untuk *Add Ads Notification* yang berfungsi untuk menambahkan iklan yang di input oleh admin, dan dapat menyiarkan notifikasi tersebut pada saat yang ditentukan admin.
3. Membuat API untuk *Update Ads Notification* berfungsi untuk memperbarui data iklan yang telah di input admin.
4. Membuat API untuk *Publish Ads Notification* berfungsi untuk mempublikasi data iklan yang telah di input oleh admin agar dapat disiarkan.
5. Membuat API untuk *Delete Ads Notification* berfungsi untuk menghapus iklan yang telah dimasukkan oleh admin sebelumnya.

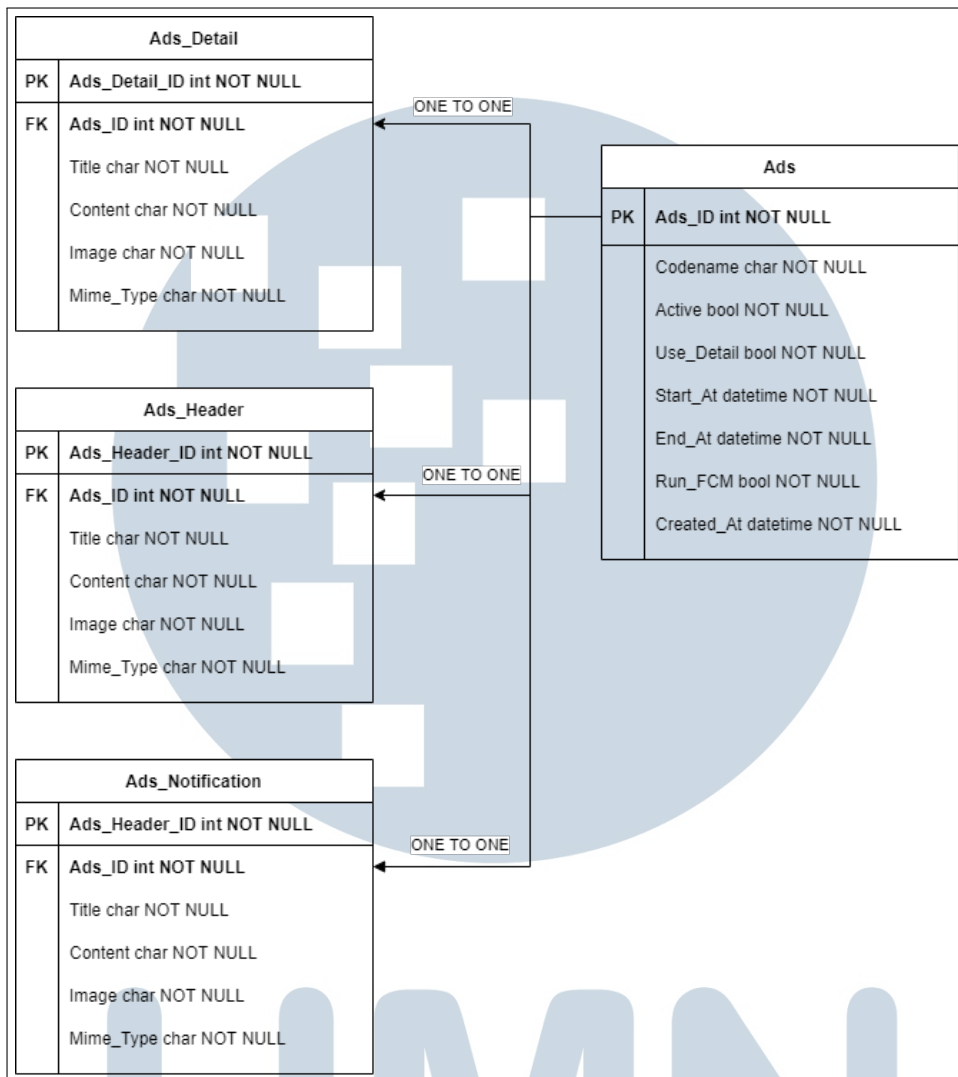
6. Membuat *Error Handling with Custom Key and Message* untuk *Ads Notification* menyediakan error handling beserta custom error key response untuk setiap kasus yang dirundingkan oleh tim Frontend dan Backend.
7. Menambahkan sistem *Logging* menggunakan *Webhook* yang tersambung dengan *Slack* yang berfungsi sebagai logger untuk melihat admin yang bertanggung jawab atas request yang dibuat.
8. Membuat *Unit Testing Script* untuk Semua API yang ada dalam *Ads* serta *Dashboard* yang berfungsi untuk testing setiap endpoint API yang sudah ada beserta kasus - kasus yang dapat terjadi.
9. Membuat Dokumentasi untuk *Ads Notification* sesuai dengan format untuk digunakan oleh tim lain.
10. Melakukan *debugging* jika terjadi *error* atau *bug* yang dilaporkan oleh rekan kerja magang atau *FrontEnd*
11. Melakukan *Maintenance*

### 3.4 Proses Pelaksanaan

#### 3.4.1 Pemahaman Alur Proyek

Proses awal yang dilakukan oleh peserta magang saat masuk ke dalam proyek adalah memahami konsep dasar serta memahami data *flow* dan arsitektur tabel dengan tujuan memahami apa yang telah dibuat dan apa yang belum dan akan dibuat. Berikut merupakan Diagram *Entity Relational Diagram* dari tabel *Database PostgreSQL 3.1*

U N I V E R S I T A S  
M U L T I M E D I A  
N U S A N T A R A



Gambar 3.1. Struktur ERD Ads Notification PT Pharos Indonesia

### 3.4.2 Diskusi Fitur Baru

Dilakukannya *briefing* dan tanya jawab yang rutin dalam 1 - 3 hari pertama saat masuk ke dalam proyek CenturyNet-Admin. Setelah memahami alur pekerjaan, peserta magang membagi tugas yang diberikan oleh atasan. Tugas utama yang diberikan adalah mengembangkan fitur baru *Fire-base Cloud Messaging* untuk notifikasi Iklan (*Ads*) dalam proyek tersebut.

### 3.4.3 Pembuatan API Basic

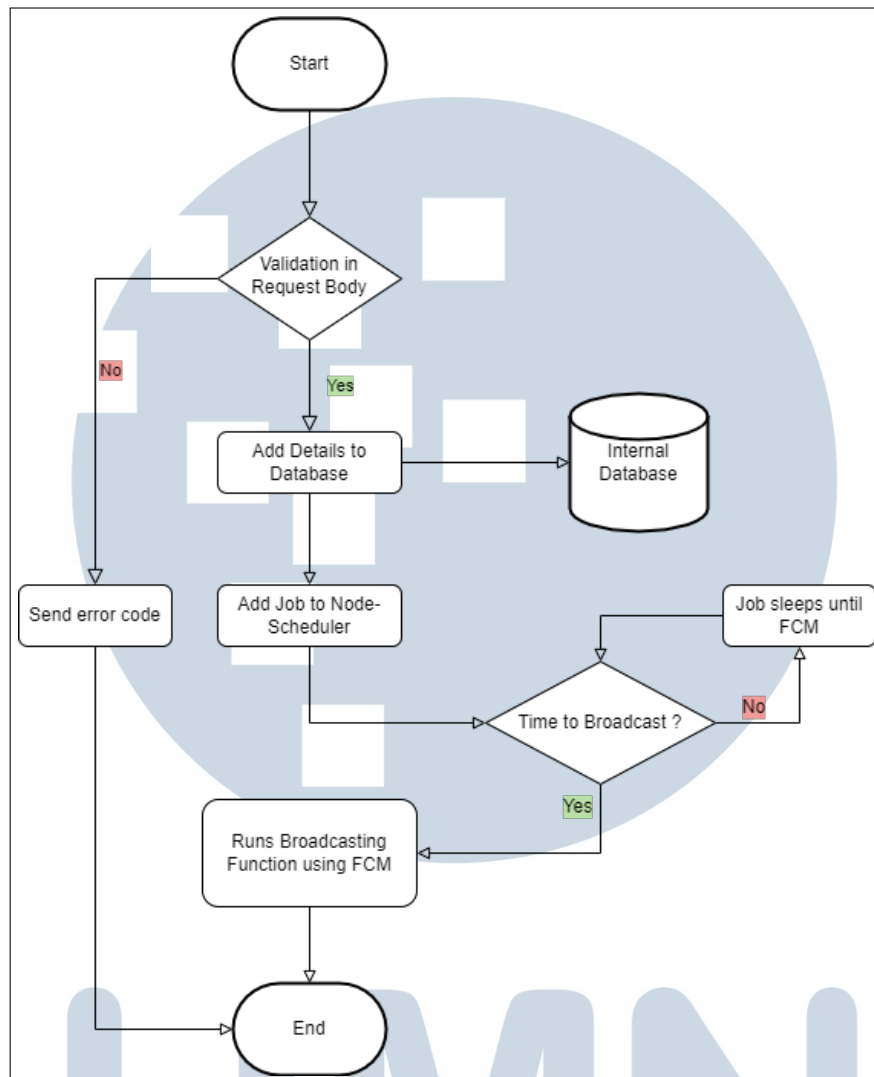
Proses pertama adalah untuk membuat CRUD dasar untuk rute *Ads* yang dilakukan menggunakan *Node.js* menggunakan library *Express.js*.

Proses pembuatan CRUD pada rute API ini sudah termasuk proses validasi data yang di ekspektasikan oleh Front-End dan juga dilengkapi dengan *basic Error Codes* untuk menanggulangi error yang di deteksi.

#### **3.4.4 Planning Node-Scheduler dan Firebase Cloud Messaging**

Setelah menyelesaikan CRUD dasar untuk API ads, masuk ke dalam perancangan fitur Posting Ads yang dimulai dengan menerapkan *Time-based Scheduling* serta konfigurasi Firebase dalam Proyek. Tujuannya menggunakan *Time-Based Scheduling* adalah untuk mengirim pesan notifikasi pada tanggal yang ditentukan oleh Admin. Untuk itu digunakan library Node-Scheduler, Node-Scheduler akan membuat *worker* yang akan aktif saat waktu yang ditentukan oleh programmer dan saat aktif maka *worker* akan menjalankan fungsi yang telah ditentukan oleh programmer yaitu menyiarkan pesan notifikasi kepada user - user yang menggunakan aplikasi baik mobile maupun web. Grafik 3.2 ini menjelaskan alur pekerjaan *Add Ads* dalam CenturyNet-Admin.





Gambar 3.2. Struktur alur sistem *Scheduling Ads* PT Pharos Indonesia

### 3.4.5 Penyelesaian Pembuatan API

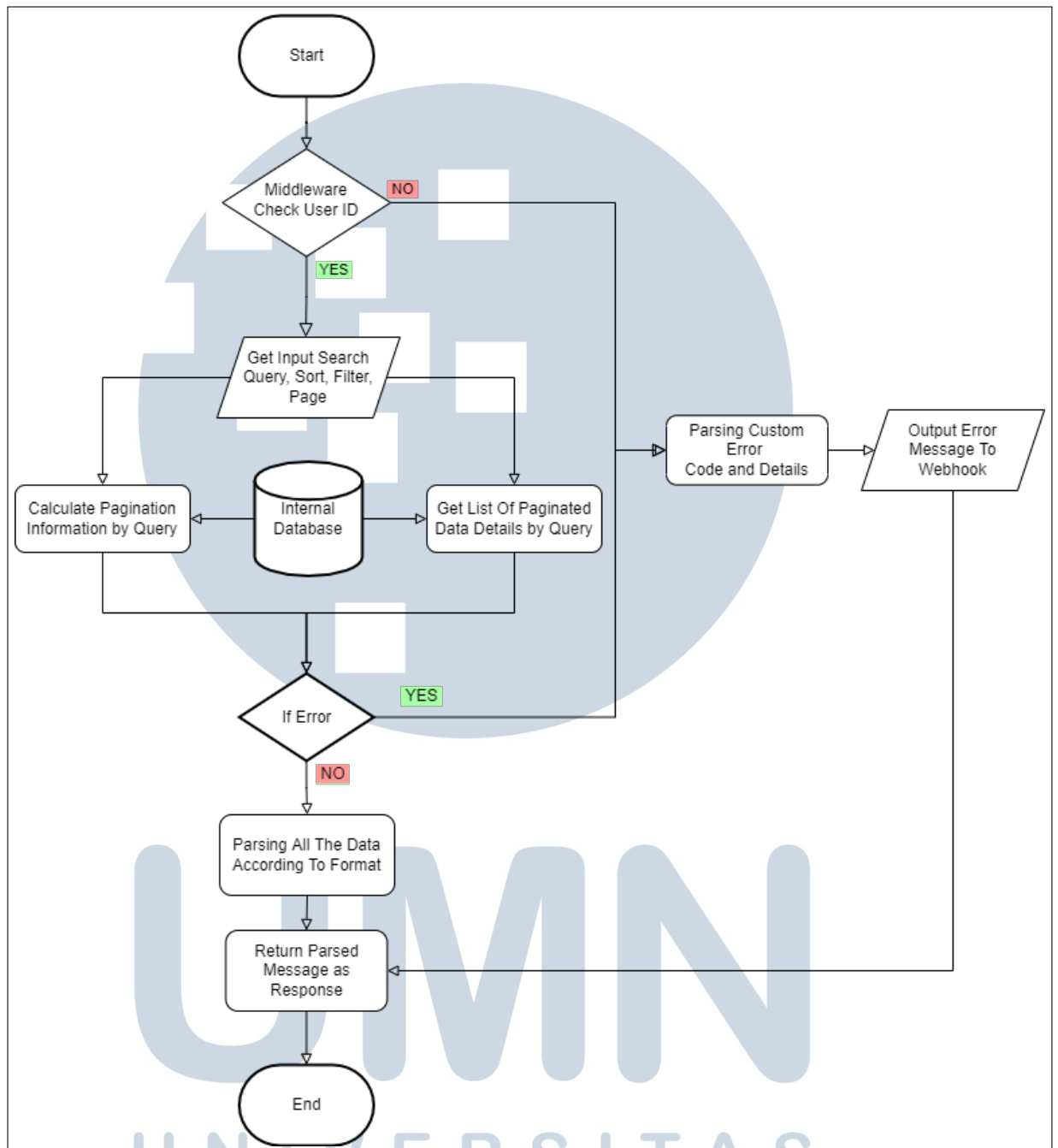
Setelah perancangan Scheduler serta konfigurasi Firebase untuk menggunakan fitur *Cloud Messaging* selesai, maka pada tahap selanjutnya adalah menyelesaikan *Endpoint API* yang dilengkapi dengan *Parameter Checking/Validation*, *Custom Error Codes*, dan implementasi rancangan Scheduler dan FCM yang telah dibuat sebelumnya. *API Endpoints* yang telah dibuat adalah :

- **Get List Of Ads Notification**

- URL : <http://localhost:8080/centurynet-admin/v1/ads/list/get>

- API yang digunakan untuk memanggil list data - data dari Ads dalam tabel database.
- API dilengkapi dengan fitur *pagination*, *query*, *filter*, dan *sorting* data.
- Dalam pemanggilan *multiple* data digunakan *parallelism* untuk mengambil data dengan waktu yang lebih efisien dengan tujuan mengatasi *high traffic endpoint*.
- Flowchart endpoint Get List Of Ads Notification :





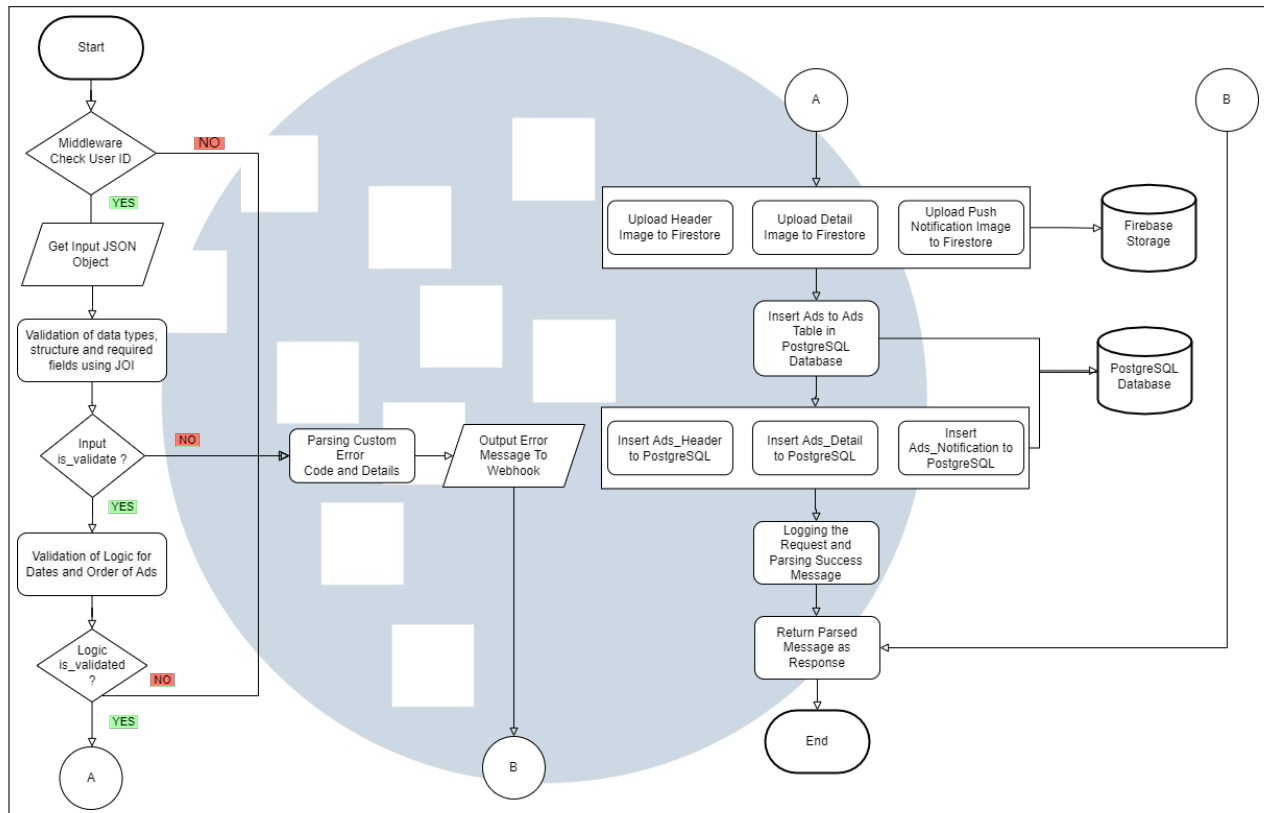
Gambar 3.3. Struktur alur Endpoint Get List of Ads Notification

Sebelum mengakses Endpoint yang dibuat harus melewati custom middleware yang telah dibuat untuk mengecek apakah user ada dan memiliki role Admin. Setelah itu memproses request dari user sesuai dengan fitur yang diminta (filter, sort, page). Lalu melakukan fetching data dari database, digunakan parallel async await untuk mengambil multiple data yang diperlukan. Setelah selesai data akan di parsing





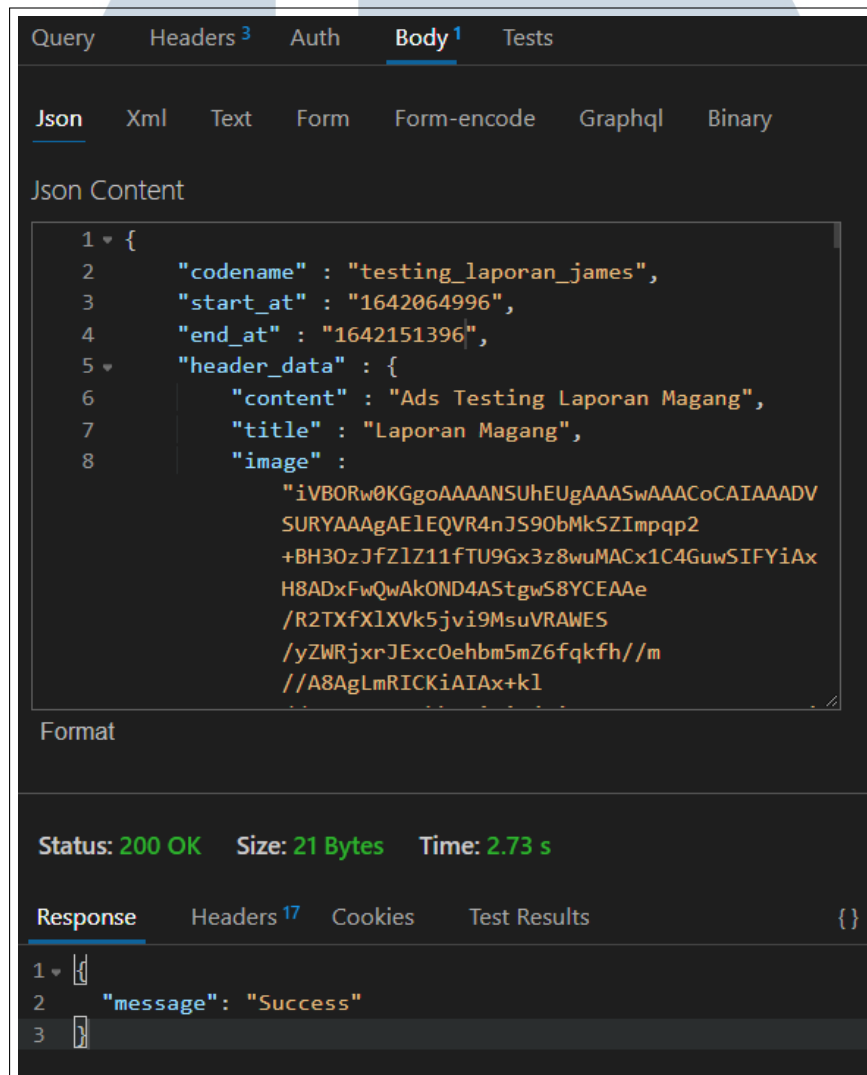
– Flowchart endpoint Add New Ads Notification :



Gambar 3.5. Struktur alur Endpoint Add Ads Notification

Sebelum mengakses Endpoint yang dibuat harus melewati custom middleware yang telah dibuat untuk mengecek apakah user ada dan memiliki role Admin. Setelah itu langkah pertama di Endpoint adalah mengambil Input yang diberikan request berbentuk JSON, dan melakukan validasi menggunakan JOI untuk mengecek data type dan limitasi input yang sesuai. Setelah melewati validasi JOI, maka lanjut kepada validasi logic yaitu melihat jika slot yang ditunjuk user telah di reserve oleh iklan lain pada tanggal yang diminta. Jika pada periode tanggal yang di request sudah di reservasi oleh iklan lain maka Endpoint akan menolak request tersebut. Jika validasi tanggal berhasil, maka server akan mengupload gambar yang di input ke dalam Firestore Storage. Terdapat tiga gambar yang di upload sesuai dengan grafik flowchart, proses berjalan dengan parallel, dan ketika semua proses upload telah berjalan sampai selesai maka url dapat diperoleh untuk langkah selanjutnya yaitu memasukan data ke dalam database Post-

greSQL (pada tabel Ads, Ads\_Detail, Ads\_Header, Ads\_Notification). Setelah semua selesai maka akan dilakukan logging dan Endpoint akan mengirimkan Success message kepada client. jika terjadi error pada saat input atau upload di dalam Endpoint, maka akan dilakukan *rollback* agar data tetap tersinkronisasi antar tabel. Berikut hasil dari API tersebut :

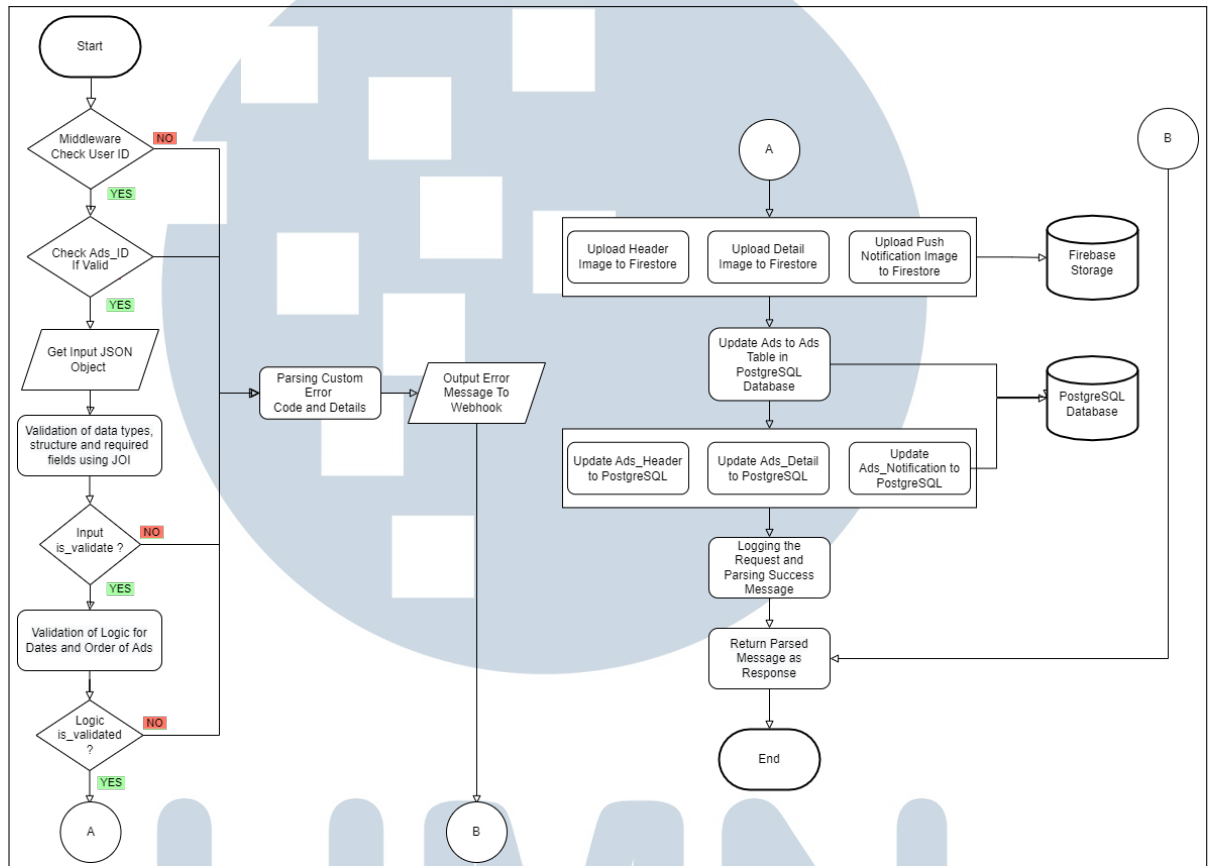


Gambar 3.6. Contoh Success Response Endpoint Add Ads

- **Update Ads Notifications**

- URL : <http://localhost:8080/centurynet-admin/v1/ads/update/:id>
- API yang digunakan untuk Update data Ads detail - detail ads yang akan ditampilkan kepada user - user sebelum di publish.

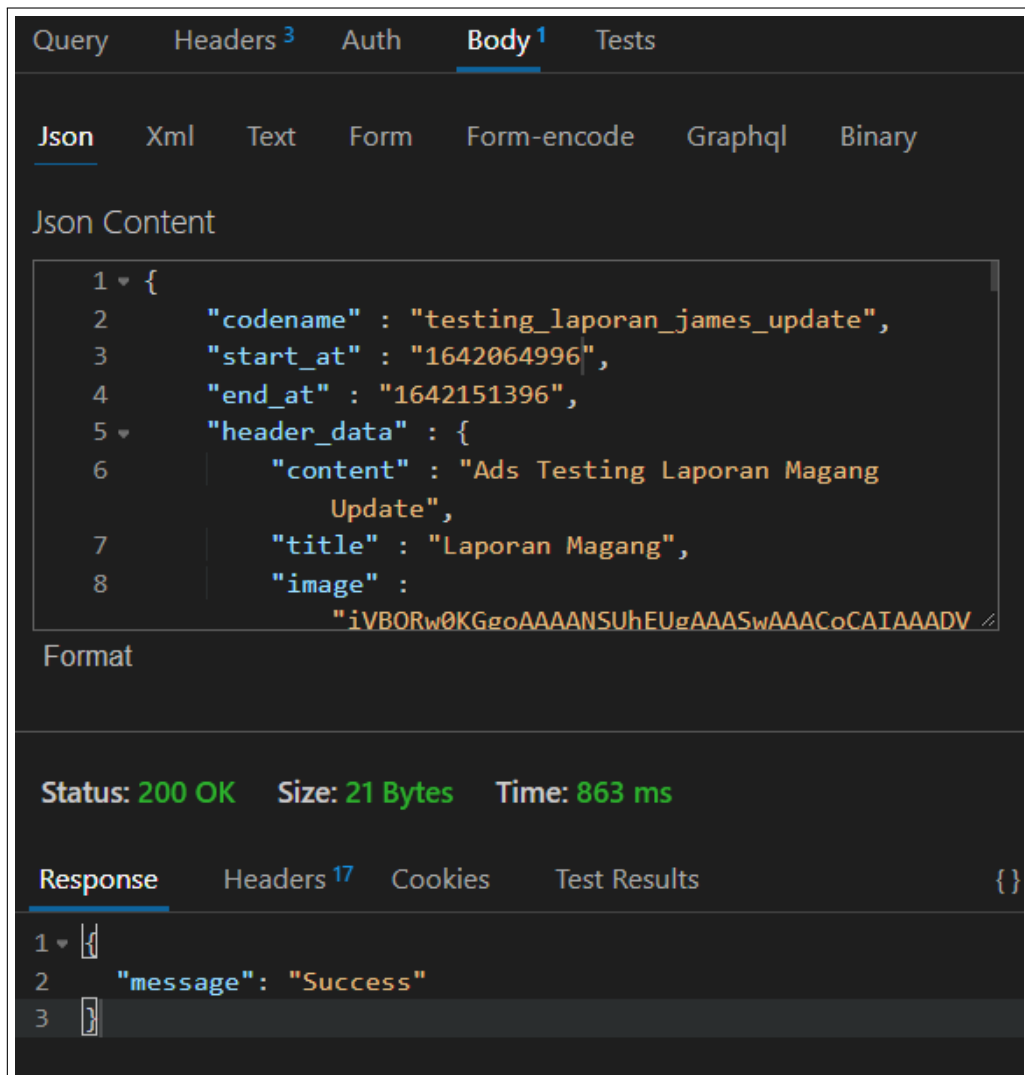
- API menggunakan Validation untuk data yang di input (penerapan Parallelism dalam pengecekan data)
- Flowchart endpoint Update Ads Notification :



Gambar 3.7. Struktur alur Endpoint Update Ads Notification

Endpoint ini hampir sama dengan Endpoint Add Ads Notification, perbedaannya terletak setelah Middleware selesai, Endpoint akan mengecek jika Ads\_ID yang ingin di update sudah terbuat atau belum, dan apabila sudah di publikasi atau belum. Sisanya sama dengan Add Ads Notification. Berikut hasil dari API tersebut :

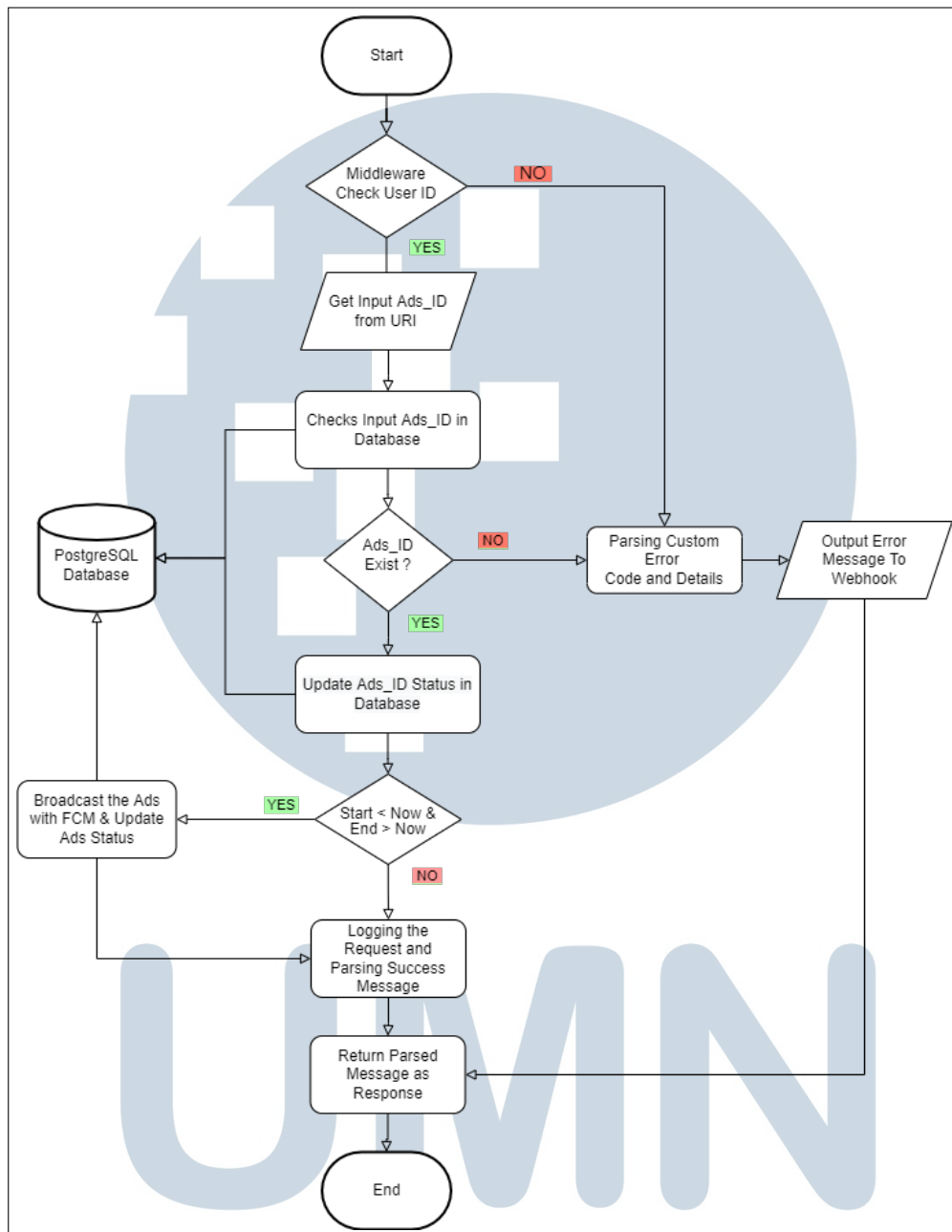
UNIVERSITAS  
MULTIMEDIA  
NUSANTARA



Gambar 3.8. Contoh Success Response Endpoint Update Ads

- **Publish Ads Notification**

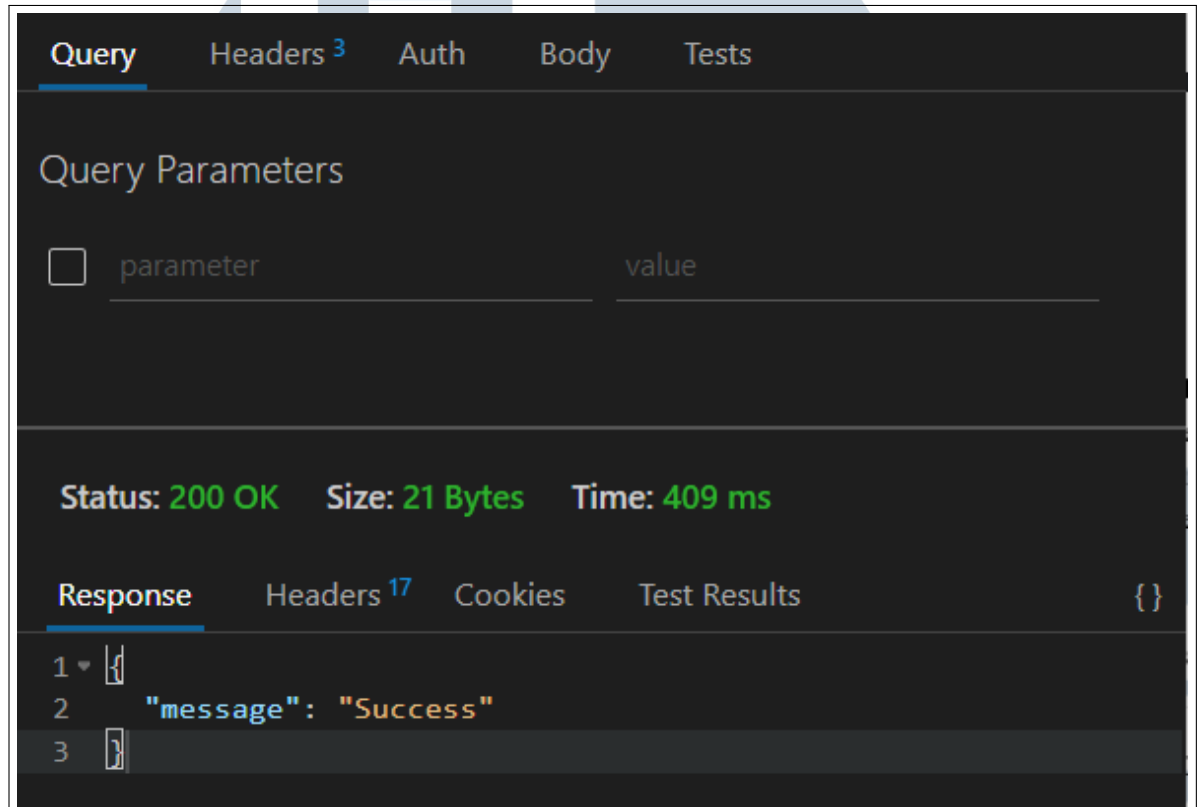
- URL : <http://localhost:8080/centurynet-admin/v1/ads/publish/:id>
- API yang digunakan untuk Publikasi yang telah di input akan ditampilkan kepada user - user. Ads yang telah di input pada API **Add New Ads Notification** tidak akan terpublikasi jika tidak menjalankan **Publish Ads Notification**.
- Flowchart endpoint Publish Ads Notification :



Gambar 3.9. Struktur alur Endpoint Publish Ads Notification

Sebelum mengakses Endpoint, dilakukan pengecekan User melalui Custom Middleware agar hanya admin yang dapat mengakses Endpoint. Selanjutnya mengambil input Ads\_ID dan mengecek jika Ads\_ID yang di input ada di dalam Database. Jika ada, maka status dari Ads\_ID tersebut akan diubah di dalam Database. Langkah berikutnya adalah mengecek jadwal dari Ads\_ID, jika Start.date sudah lewat dan End.date belum lewat maka fungsi FCM akan di eksekusi secara langsung dan

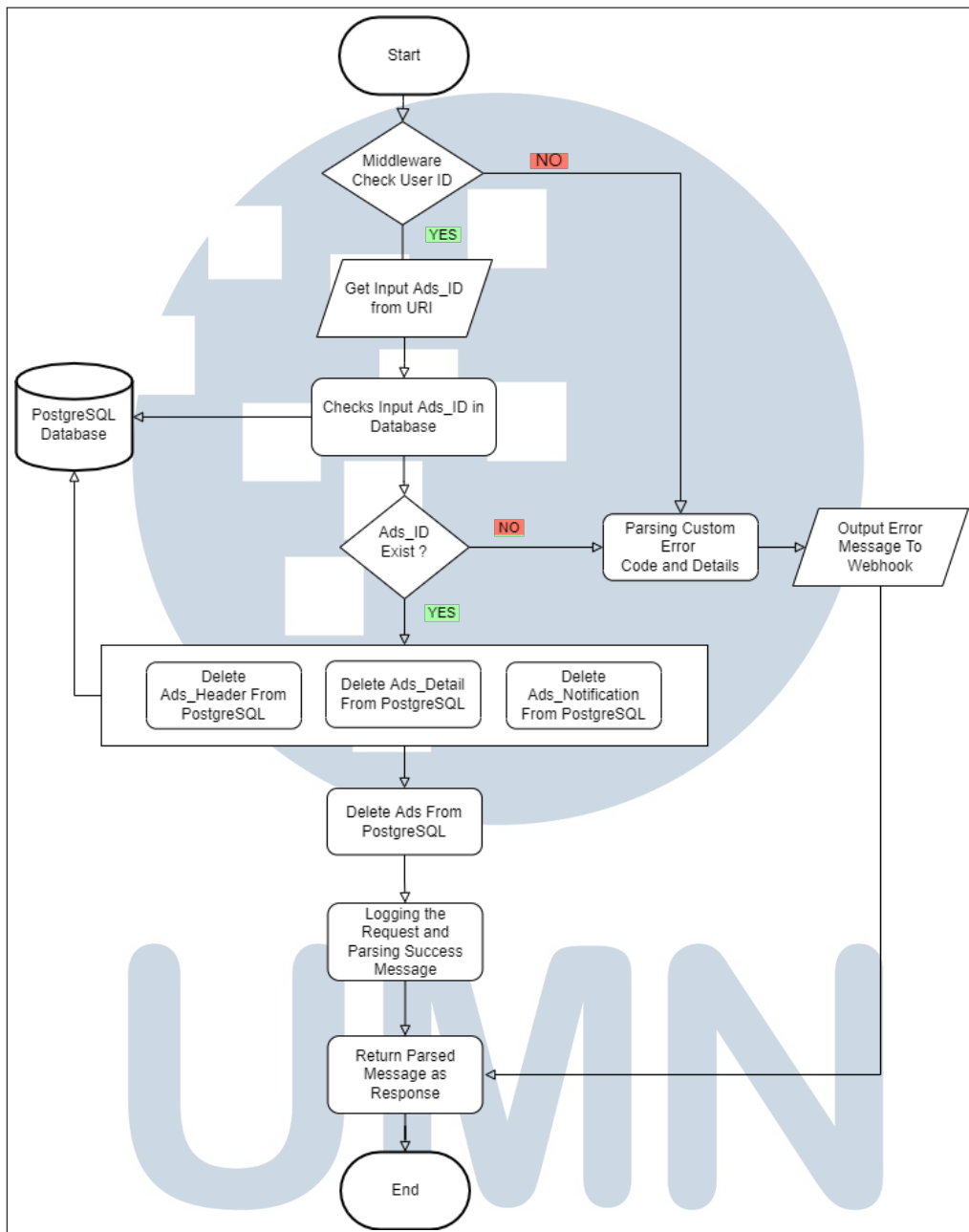
status dari Ads\_ID akan diubah. Jika sudah, maka selanjutnya adalah logging dan akan mengirimkan Success message kepada client. Jika tidak masalah dengan Star\_date dan End\_date maka langsung kepada bagian logging dan seterusnya hingga selesai. Berikut hasil dari API tersebut :



Gambar 3.10. Contoh Success Response Endpoint Publish Ads

- **Delete Ads Notification**

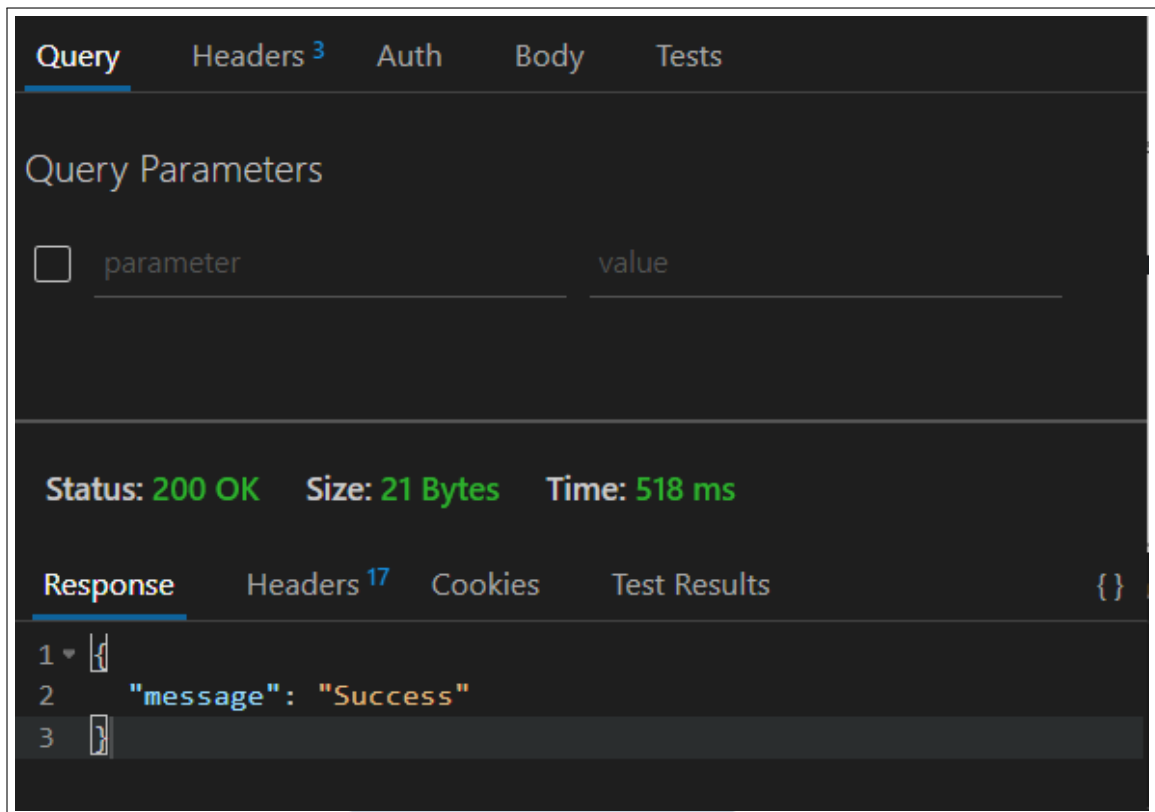
- URL : `http://localhost:8080/centurynet-admin/v1/ads/del/:id`
- API yang digunakan untuk menghapus Ads yang telah di input.
- Flowchart endpoint Delete Ads Notification :



Gambar 3.11. Struktur alur Endpoint Delete Ads Notification

Struktur endpoint ini hampir sama dengan endpoint Publish Ads Notification, perbedaannya terletak setelah pengecekan Ads.ID dari input, Endpoint akan melakukan aksi menghapus data Ads dari tabel yang berada di PostgreSQL, setelah itu akan di akhiri dengan logging serta parsing message success yang akan dikembalikan kepada user sebagai response. Berikut hasil dari API tersebut :





Gambar 3.12. Contoh Success Response Endpoint Delete Ads

### 3.4.6 Unit Testing

Setelah proses pembuatan API selesai dilakukannya Unit Testing untuk mengecek kasus - kasus yang dapat terjadi dalam Endpoint. Dalam proses ini programmer fokus kepada pencarian bug dan kelemahan dari endpoint yang telah dibuat dengan tujuan membuat Error Handling yang sesuai agar tim Frontend dapat memproses Error tersebut. Mocha.Js Framework merupakan framework yang digunakan untuk melakukan script testing. Berikut contoh dari testing tersebut :

UNIVERSITAS  
MULTIMEDIA  
NUSANTARA

```
C:\Users\Asus\Desktop\centurynet-admin>npm test

> admin-app@1.0.0 test C:\Users\Asus\Desktop\centurynet-admin
> mocha --exit

Terminate batch job (Y/N)? n

C:\Users\Asus\Desktop\centurynet-admin>npm test

> admin-app@1.0.0 test C:\Users\Asus\Desktop\centurynet-admin
> mocha --exit

[CenturyNet-Admin] getSkippedNotRunningAdsFCM
[CenturyNet-Admin] getmaxidads

GET /centurynet-admin/v1/ads/dashboard/get/main_ads/:order_number
10
[CenturyNet-Admin] GetMainAds
[CenturyNet-Admin] getNotRunningAdsFCM
[CenturyNet-Admin] getSkippedNotRunningNotifFCM
[CenturyNet-Admin] getNotRunningNotifFCM
  ✓ Test Success (505ms)
  ✓ Test Failed

GET /centurynet-admin/v1/ads/dashboard/get/queue/:order_number
[CenturyNet-Admin] GetActiveDashboardAdsList
  ✓ Test Success (55ms)
  ✓ Test Failed

GET /centurynet-admin/v1/ads/dashboard/get/ads_history/:order_number
[CenturyNet-Admin] GetHistoryDashboardAdsList
  ✓ Test Success (65ms)
  ✓ Test Failed
```

Gambar 3.13. Contoh Unit Testing Script Mocha.js

### 3.4.7 Dokumentasi

Dokumentasi merupakan tahap terakhir dalam proyek CenturyNet Admin Ads. Pada tahap ini programmer membuat dokumentasi Path, Requirements, dan Responses dari setiap API yang telah dibuat (Header, Param, Query, Body, Error Key). Dokumentasi dibuat dalam format yang telah diberi oleh supervisor, dan dibuat dalam Google Docs.

## 3.5 Time Table

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	Pemahaman alur proyek dan diskusi spesifikasi fitur baru
2	Pembuatan API Basic, Planning Scheduler, dan FCM
3-4	Penyelesaian Pembuatan API
5	Pembuatan Script Unit Testing dan dokumentasi

### 3.6 Kendala dan Solusi yang Ditemukan

#### **Kendala :**

1. Pada bagian *scheduler*, jika program berhenti baik untuk *maintenance* atau karena suatu error maka *Jobs* yang sudah terdaftar di dalam *scheduler* akan hilang.
2. Dokumentasi Firebase Admin SDK yang kurang lengkap.

#### **Solusi :**

1. Menyimpan status *Jobs* tersebut dalam record database sebagai penanda bahwa job sudah dijalankan atau belum. Pengecekan *Job* yang sudah atau belum dijalankan akan dilaksanakan setiap kali program pertama kali dijalankan.
2. Mencari diskusi tentang Firebase Admin di forum, serta mencari informasi melewati Youtube.

U N I V E R S I T A S  
M U L T I M E D I A  
N U S A N T A R A