# BAB 3
# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Organisasi

Pelaksanaan kerja magang dilakukan pada posisi *Software Engineer* di PT Tokopedia. Kerja magang dilakukan di bawah bimbingan oleh Lawrence Supian selaku Leader di Tim Asuransi dan Investasi pada divisi *Financial Technology* di Tokopedia. Di tim ini berisikan kurang lebih 10 anggota yang terdiri dari *Product Manager*, *Software Engineer* dan *Web Platform Engineer*. Tim ini fokus pada 3 produk utama di Tokopedia yaitu Emas, Reksa Dana, dan Asuransi.

## 3.2 Tugas yang Dilakukan

Dalam melaksanakan kerja magang, tugas utama yang dilakukan adalah mengembangkan *API* dari sisi *Back End* untuk memenuhi semua kebutuhan dalam pengembangan sistem internal tools untuk tim *Financial Technology* khususnya modul Asuransi, Reksa dana, dan Kamus Asuransi. Dalam implementasinya, terdapat beberapa technology utama yang digunakan diantaranya:

- Bahasa Pemrograman Go
- Database MySQL
- Redis
- Docker Container
- Postman

Selama proses kerja magang, terdapat beberapa modul yang dikerjakan untuk memenuhi kebutuhan pengembangan, diantaranya:

- Modul Reksa dana

  Pada modul ini yang dikerjakan adalah bagian manajemen pendapatan, dimana di dalamnya terdapat beberapa fitur seperti menampilkan semua data, filter data, dan ekspor data ke dalam format excel.

- Modul Asuransi

  Pada modul ini yang dikerjakan adalah bagian manajemen pendapatan, dimana di dalamnya terdapat beberapa fitur seperti menampilkan semua data, filter data, dan ekspor data ke dalam format excel.

- Modul Kamus Asuransi Tokopedia

  Pada modul ini yang dikerjakan adalah bagian manajemen *post*, dimana di dalamnya terdapat beberapa fitur seperti menampilkan, menambah, mengubah, dan menghapus data.

- Modul Asuransi Penerbangan

  Pada modul ini yang dikerjakan adalah bagian konfigurasi asuransi penerbangan, dimana di dalamnya terdapat beberapa fitur seperti menampilkan semua data, filter data, menambah, mengubah, dan menghapus data.

## 3.3 Uraian Pelaksanaan Magang

Kerja magang yang dilakukan di PT Tokopedia berlangsung selama 18 minggu sampai selesai dibuatnya laporan magang ini. Namun proyek pengembangan sistem internal tools v2 pada modul asuransi, reksa dana, dan kamus teknologi keuangan ini dilakukan dalam waktu 7 minggu pertama. Berikut adalah uraian dari kegiatan yang dikerjakan selama kerja magang. Pelaksanaan kerja magang diuraikan pada Tabel 3.1, 3.2, 3.3 dan 3.4.

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

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 1 | • Masa *Pre-Onboarding* dan mengenal lingkungan kerja di Tokopedia<br>• Belajar mandiri terkait bahasa pemrograman Go |
| 2 | • Masa *Pre-Onboarding* dan menyelesaikan *Challenge* #NakaMakeItBetter bersama tim yang telah ditentukan<br>• Belajar mandiri terkait *Clean Architecture* di bahasa pemrograman Go |
| 3 | • Masa *Onboarding* dan mulai dikenalkan dengan anggota tim divisi<br>• *Setup working environment* dan *tools-tools* yang digunakan di tim |
| 4 | • Mempelajari dan mendalami *repository* Reksa Dana<br>• Mengembangkan *API* untuk modul Reksa Dana |
| 5 | • Mempelajari dan mendalami *repository* asuransi<br>• Mengembangkan *API* untuk modul Asuransi |
| 6 | • Mempelajari dan mendalami *repository* kamus asuransi tokopedia<br>• Mengembangkan *API* untuk modul Kamus Asuransi Tokopedia |
| 7 | • Mempelajari dan mendalami *repository* asuransi penerbangan<br>• Mengembangkan *API* untuk modul Asuransi Penerbangan |

Tabel 3.2. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan 2)

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 8 | • Menambahkan log aktivitas untuk data PII (Personally Identifiable Information)<br>• Migrasi *monitoring system* dari datadogh ke new relic untuk *repository* reksa dana dan pembelian proteksi<br>• Membuat *popup handler* untuk migrasi emas dari pegadaian ke pluang untuk pengguna yang memenuhi syarat |
| 9 | • Meeting untuk mendiskusikan Migrasi Emas dari sistem OMS ke TP<br>• Meeting terkait code scanner untuk mencegah adanya *XSS (Cross Site Scripting)*<br>• *Sprint requirement and planning meeting*<br>• Investigasi masalah *summarize* pendapatan harian di produk Emas |
| 10 | • Menyelesaikan code scanner untuk produk emas dan asuransi digital<br>• Memperbaiki bug intools *flight insurance*<br>• Mengembangkan *API* baru untuk flow checkout init pembelian produk emas |
| 11 | • Testing dan Perbaikan pada *API* checkout init produk emas<br>• Mengembangkan *API* webhook yang akan digunakan oleh tim lain ketika pembayaran produk emas telah selesai / dibatalkan<br>• *Sprint requirement and planning meeting* |

Tabel 3.3. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan 3)

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 12 | • Testing dan perbaikan pada *API* webhook produk emas<br>• Mengembangkan *API* untuk alur berlangganan emas di produk emas<br>• Mengembangkan *API* untuk alur *refund* emas di produk emas<br>• Discussion meeeting untuk integrasi klaim melalui webview dengan partner |
| 13 | • Testing dan perbaikan pada *API* untuk *subscription* dan *refund* produk emas<br>• Mempelajari *repository* pembelian proteksi asuransi dan klaim asuransi di Tokopedia<br>• Mengembangkan *API* webhook untuk integrasi dengan partner ketika klaim asuransi<br>• *Sprint requirement and planning meeting* |
| 14 | • *End to End Testing* untuk semua alur pembelian produk emas menggunakan flow baru<br>• *Discussion meeting* dengan partner untuk integrasi token portal klaim asuransi<br>• *Debug timeout error* di intools *insuretech transaction* |

Tabel 3.4. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang (lanjutan 4)

| Minggu Ke - | Pekerjaan yang dilakukan |
|---|---|
| 16 | • Mengembangkan buy confirm alert monitoring dengan new relic<br>• Memperbaiki bugs ketika *testing flow* baru pembelian produk emas<br>• Membuat rollout fitur menggunakan *feature flags*, *rollence*, dan *whitelist* untuk flow baru pembelian emas |
| 17 | • Testing support untuk klaim asuransi melalui partner<br>• Testing support untuk pembelian emas melalui flow baru<br>• Menambahkan dan menyesuaikan encryption data dengan partner menggunakan metode RSA dan AES<br>• *Sprint requirement and planning meeting* |
| 18 | • Testing support untuk klaim asuransi melalui partner<br>• Mengubah flow pembelian emas untuk pembelian pertama dan integrasinya dengan partner |

Pada Tabel 3.1, 3.2, 3.3, dan 3.4 di atas merupakan uraian singkat pelaksanaan kerja magang yang dilakukan di Tokopedia. Berikut adalah penjelasan mendetail untuk 7 minggu yang awal yang dilakukan untuk mengembangkan proyek sistem internal tools pada modul asuransi, reksa dana, dan kamus di Tokopedia:

- Pada minggu pertama adalah masa *Pre-Onboarding* untuk semua partisipan magang di Tokopedia, terdapat kurang lebih 90 partisipan dari berbagai kampus dan jurusan. Disini kami dikenalkan satu sama lain dan dijelaskan terkait alur masa magang di Tokopedia. Tidak hanya itu, kami juga disambut langsung oleh *Co-Founder* Tokopedia yaitu Pak Leontinus Alpha Edison.
- Pada minggu kedua, semua partisipan magang dibagi menjadi 10 tim dengan komposisi tim yang acak dan kemudian setiap tim ditantang

untuk menyelesaikan *challenge* #NakaMakeItBetter yaitu tantangan untuk memberikan ide inovasi untuk kemajuan Tokopedia.

- Pada minggu ketiga adalah masa *Onboarding* dimana semua partisipan sudah mulai tergabung dalam tim kecil sesuai divisi magang masing-masing. Disini kami dikenalkan dengan tim dan aturan-aturan di Tokopedia, terkait privasi, penggajian, dan aturan dan sistem yang digunakan. Selain itu, di dalam tim kami diberikan informasi terkait tugas yang akan dikerjakan dan sistem manajemen tugas di tim ini.
- Pada minggu keempat, tugas yang diberikan adalah mengembangkan *API* untuk modul Reksa Dana. Tidak hanya sekedar API saya, tetapi dalam implementasinya banyak yang harus dipersiapkan seperti *Unit Test* yang harus selalu dibuat untuk setiap kasus kebutuhan dan untuk setiap baris kode yang dibuat. Setelah semua selesai, akan ada sesi *review* kode dari para senior sehingga hasil kode yang dihasilkan benar-benar bagus dan bersih. Jika semua sudah terpenuhi dan kode yang dibuat sudah disetujui oleh para senior, maka kode yang dibuat akan di *merge* dengan *repository* di master dan *staging* yang kemudian akan dilanjutkan ke fase tes *API endpoint* setelah di *deploy* ke *staging*.
- Pada minggu kelima, tugas yang diberikan adalah mengembangkan *API* untuk modul Asuransi. Seperti pada minggu sebelumnya, disini akan dibuat *Unit Test* untuk setiap baris kode. Dan kemudian akan melalui fase *review* oleh senior dan tes *API endpoint* sebelum tugas dinyatakan telah selesai.
- Pada minggu keenam, tugas yang diberikan adalah mengembangkan *API* untuk modul Kamus Asuransi Tokopedia. Seperti pada minggu sebelumnya, disini akan dibuat *Unit Test* untuk setiap baris kode. Dan kemudian akan melalui fase *review* oleh senior dan tes *API endpoint* sebelum tugas dinyatakan telah selesai.
- Pada minggu ketujuh, tugas yang diberikan adalah mengembangkan *API* untuk modul Asuransi Penerbangan. Seperti pada minggu sebelumnya, disini akan dibuat *Unit Test* untuk setiap baris kode. Dan kemudian akan melalui fase *review* oleh senior dan tes *API endpoint* sebelum tugas dinyatakan telah selesai.

### 3.3.1 Proses Pelaksanaan

Dalam proses perancangan dan pengembangan sistem internal tools Tokopedia, diperlukan perangkat pendukung baik software maupun hardware. Berikut ini adalah software dan hardware yang digunakan selama proses pelaksanaan kerja magang:

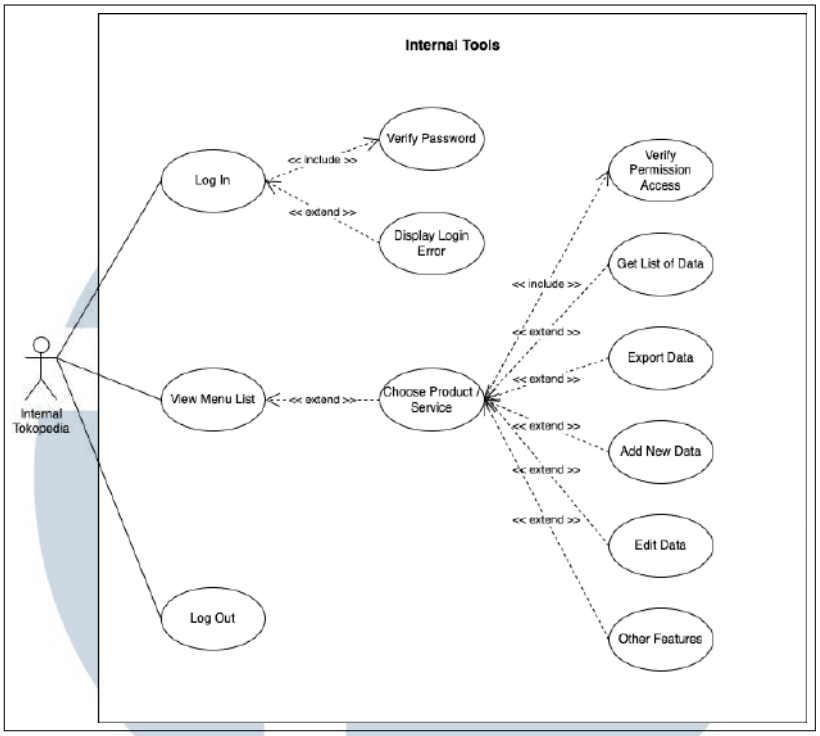
Hardware:

- Laptop Macbook Pro M1 2020
  - Sistem Operasi: Mac OS
  - Chip : Apple M1
  - RAM: 16gb
  - Memory: 512gb

Software:

- MySQL v5
- Go Language v1.17
- Redis v6.2.5
- Docker
- Visual Studio Code
- Google Chrome v90.0.4430.72
- MySQL Workbench
- Postman

Secara keseluruhan, sistem internal tools versi 2 Tokopedia ini dibuat untuk kebutuhan semua bagian internal di Tokopedia, tidak hanya divisi *Financial Technology* saja, sehingga tentunya dibutuhkan *system design* sebagai acuan utama pembuatan sistem ini agar sistem yang dibuat dapat digeneralisasi dan tidak berbeda-beda antar setiap tim maupun divisi. Berikut adalah *general system design* dari sistem internal tools Tokopedia yang didefinisikan dalam bentuk *use case diagram* pada (Gambar 3.1) :




Gambar 3.1. *Use Case Diagram Internal Tools* Tokopedia

Pada proses pelaksanaan, dibagi menjadi 4 modul besar, yaitu:

1. Modul Reksa Dana dan Asuransi

   Karena fitur yang dikerjakan pada modul Reksa Dana dan asuransi adalah manajemen pendapatan dan memiliki banyak kesamaan, maka akan didefinisikan secara bersama.

   (a) Persyaratan dan Perancangan

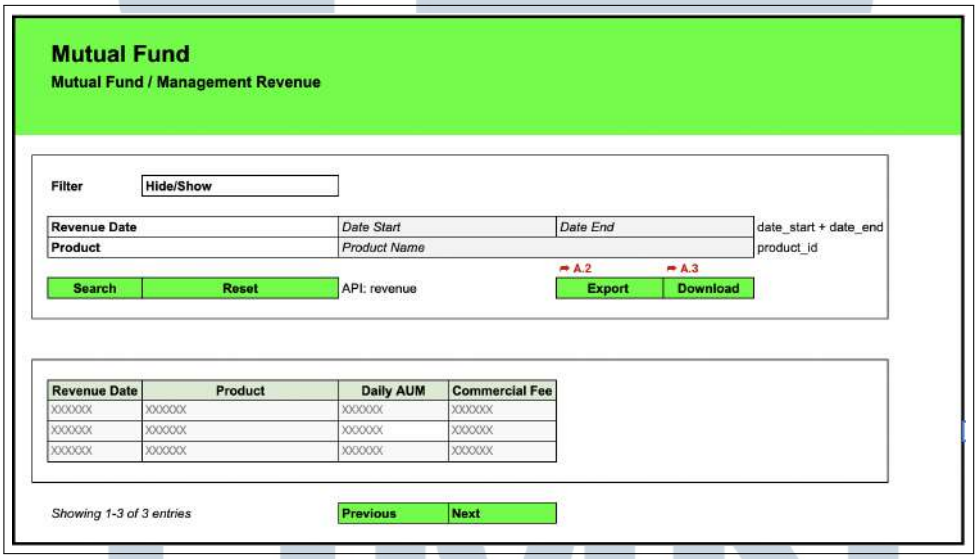
   Dalam pengembangan kedua modul ini, bagian yang ingin difokuskan adalah halaman manajemen pendapatan. Terdapat beberapa persyaratan dalam pengembangan halaman ini diantaranya:

   - Menggunakan bahasa pemrograman Go
   - Menggunakan basis data MySQL
   - Menggunakan sistem caching Redis
   - Menggunakan metode penulisan *Clean Code*
   - Membuat *Unit Test* untuk setiap baris kode

   Terdapat beberapa fitur yang diharapkan pada halaman manajemen pendapatan, diantaranya:

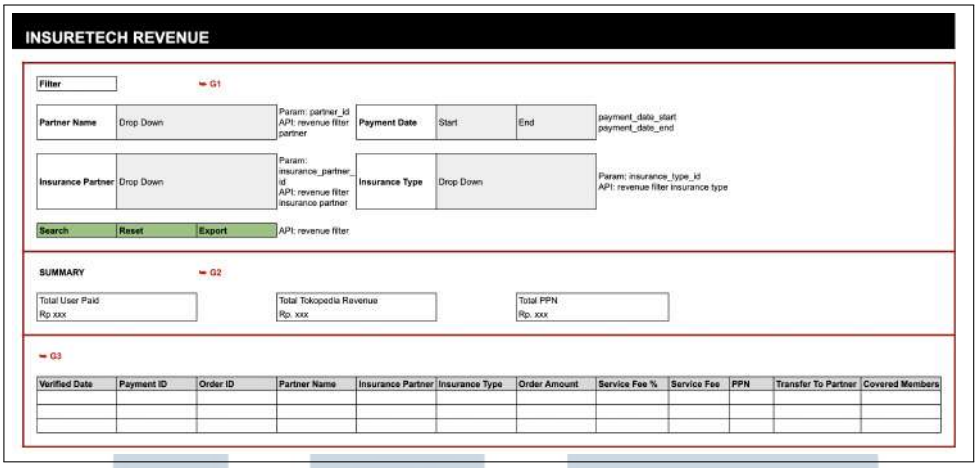
- Menampilkan semua data dengan fitur *limit, paging, sort, dan filter*
- Menampilkan ringkasan dari total pendapatan bagi pengguna yang mempunyai akses
- Menampilkan *filter select option* berdasarkan produk
- Melakukan ekspor data yang telah ditampilkan pada fitur pertama, dan dapat melihat status ekspor serta dapat membatalkan proses ekspor data

Selain persyaratan di atas, terdapat tahap perancangan, dimana pada modul Reksa Dana telah disediakan *mock up* gambaran halaman terkait halaman manajemen pendapat seperti terlampir pada Gambar 3.2.




Gambar 3.2. Mock Up Modul Reksa Dana Halaman Manajemen Pendapatan

Pada modul asuransi, juga telah disediakan *mock up* gambaran halaman terkait halaman manajemen pendapatan seperti terlampir pada Gambar 3.3.




Gambar 3.3. Mock Up Modul Asuransi Halaman Manajemen Pendapatan

Pada Gambar 3.2 dan 3.3 telah ditampilkan gambaran dari halaman yang nantinya akan diimplementasikan oleh tim *front end*. Dari *mock up* tersebut kemudian didefinisikan *API endpoint* yang akan dikembangkan, diantaranya:

i. *Endpoint* /revenue
   Bertujuan untuk menampilkan semua data dengan fitur *limit, paging, sort, dan filter*

ii. *Endpoint* /revenue/filter
   Bertujuan untuk menampilkan data produk yang dapat digunakan untuk filter pada *endpoint* /revenue

iii. *Endpoint* /revenue/export
   Bertujuan untuk menginisialisasi/memulai proses ekspor data menjadi excel

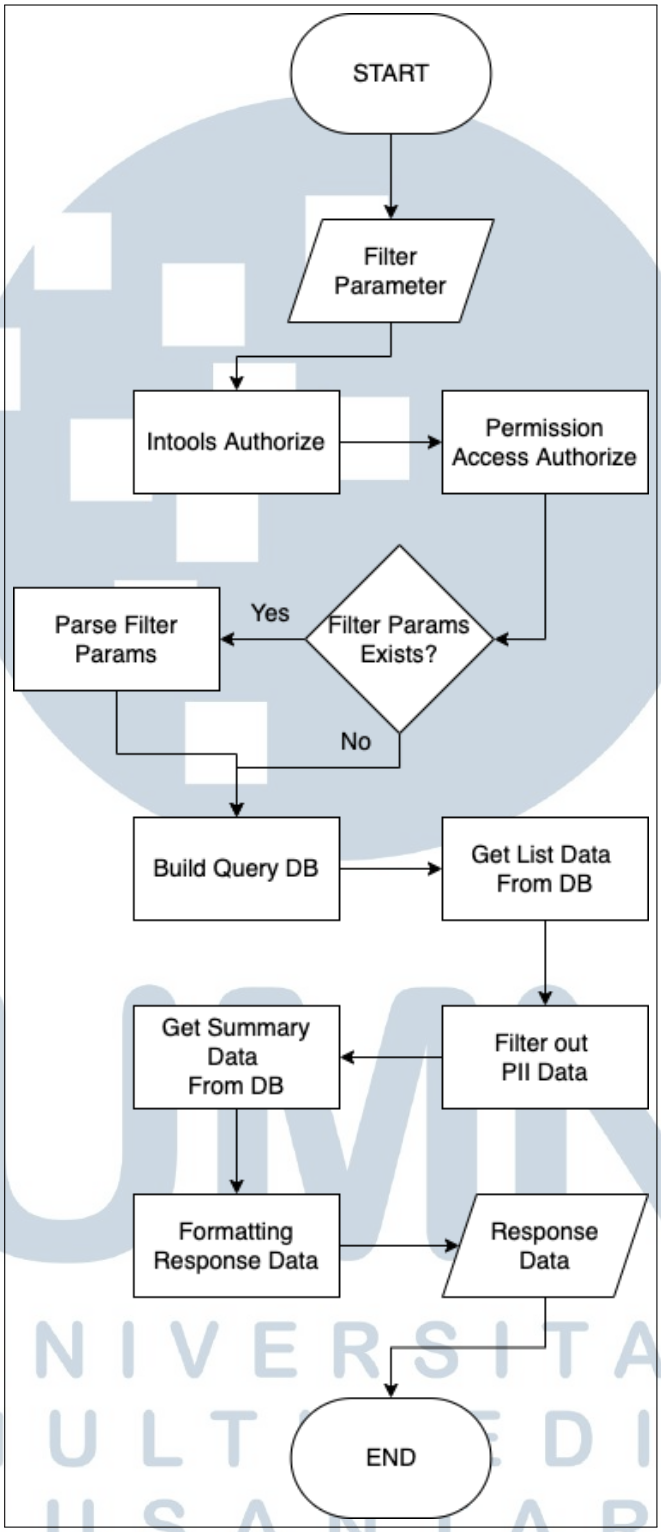
iv. *Endpoint* /revenue/export/status
   Bertujuan untuk melihat status dari proses ekspor, jika proses ekspor sudah selesai, maka akan muncul *link download file* dengan format excel

v. *Endpoint* /revenue/export/cancel
   Bertujuan untuk melakukan pembatalan pada proses ekspor jika status ekspor belum selesai

(b) Implementasi

i. *Endpoint* /revenue

Tujuan utama dari *endpoint API* ini adalah untuk mendapatkan daftar data *revenue* pembelian Reksa Dana dan asuransi di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk mendapatkan data *revenue* Reksa Dana dan asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.4.




Gambar 3.4. *Flowchart API /revenue* di Modul Reksa Dana dan Asuransi

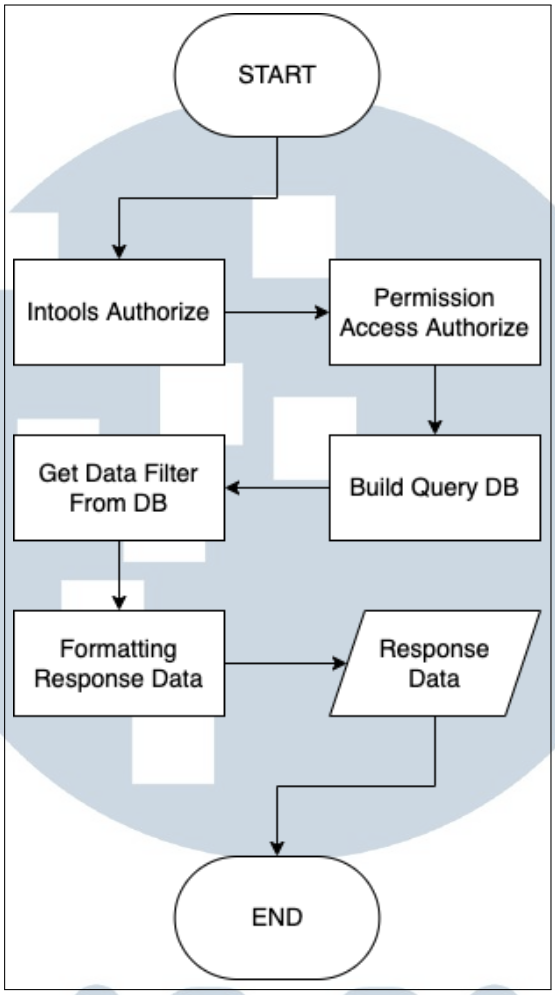
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.5.

```
type Response struct {
    Data struct {
        Table struct {
            Total     int `json:"total"`
            TotalPage int `json:"total_page"`
            Page      int `json:"page"`
            Limit     int `json:"limit"`
            Rows      []struct {
                ID          int     `json:"id"`
                Date        string  `json:"date"`
                Amount      float64 `json:"amount"`
                FeeAmount   float64 `json:"fee_amount"`
                Status      int     `json:"status"`
                Unit        float64 `json:"unit"`
                ProductID   int     `json:"product_id"`
                ProductCode string  `json:"product_code"`
                ProductName string  `json:"product_name"`
            } `json:"rows"`
        } `json:"table"`
        Cards []interface{} `json:"cards"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.5. *Struktur data response API /revenue di Modul Reksa Dana dan Asuransi*

ii. *Endpoint* /revenue/filter

Tujuan utama dari *endpoint API* ini adalah untuk mendapatkan daftar data filter yang dapat digunakan untuk menyaring / filter data sesuai kebutuhan. Secara sederhana, alur jalannya program yang dibuat untuk mendapatkan data filter *revenue* ini didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.6.



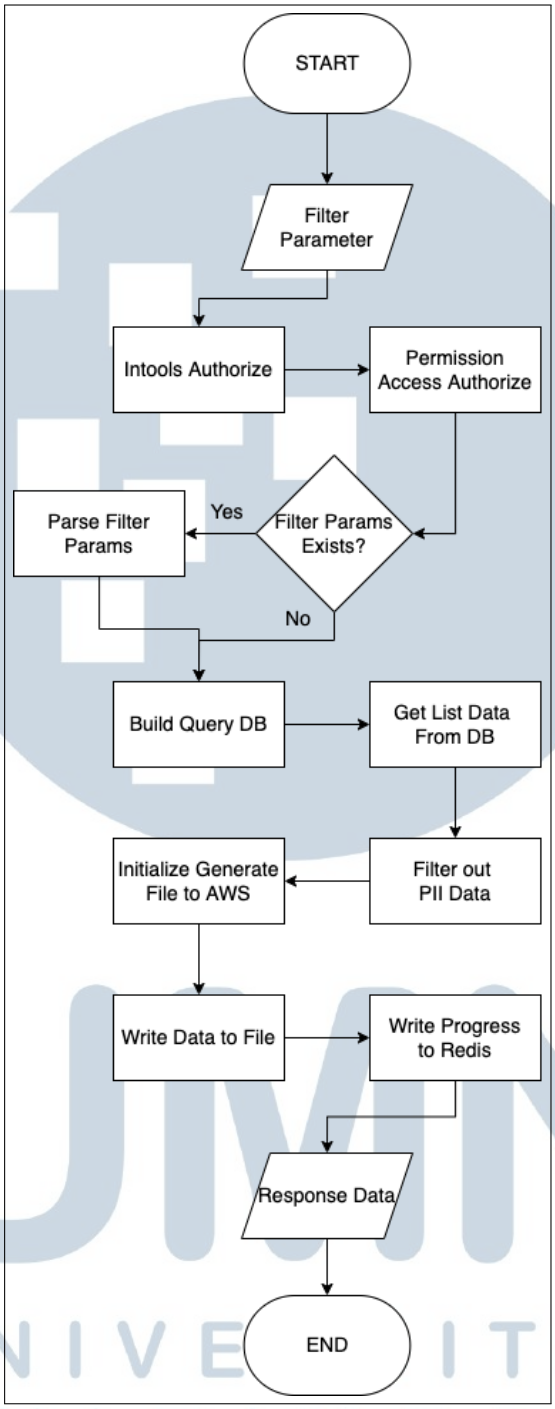

Gambar 3.6. *Flowchart API /revenue/filter* di Modul Reksa Dana dan Asuransi

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.7.


Gambar 3.7. *Struktur data response API /revenue/filter di Modul Reksa Dana dan Asuransi*

iii. *Endpoint* /revenue/export

Tujuan utama dari *endpoint API* ini adalah untuk melakukan ekspor daftar data ke dalam format excel (.xlsx), pada *endpoint* ini juga dilengkapi fitur untuk melakukan filter data. Secara sederhana, alur jalannya program yang dibuat untuk melakukan ekspor data ini didefinisikan dengan menggunakan *flowchart diagram* pada Gambar 3.8




Gambar 3.8. *Flowchart API /revenue/export* di Modul Reksa Dana dan Asuransi

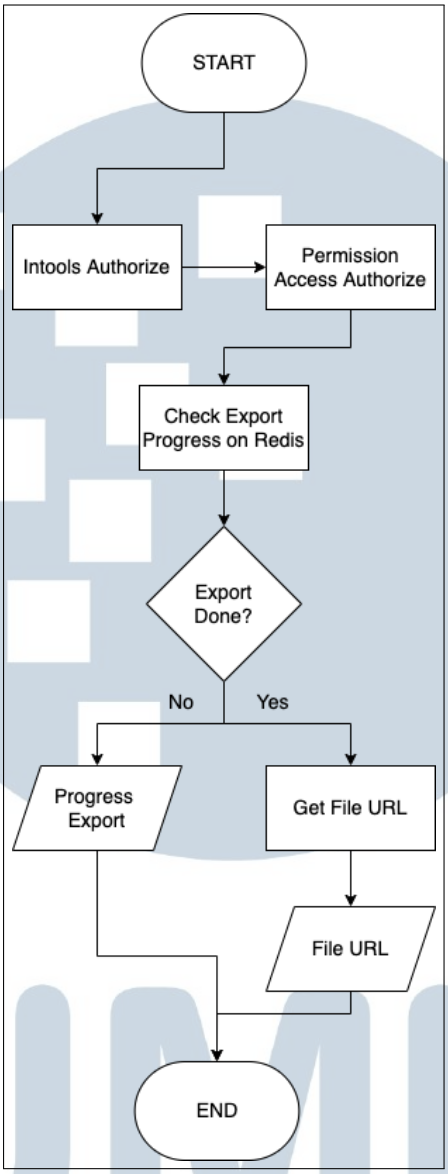
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.9.

```
type Response struct {
    Data struct {
        PresignedURL string `json:"presigned_url"`
        Status       string `json:"status"`
        Caption      string `json:"caption"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.9. *Struktur data response API /revenue/export di Modul Reksa Dana dan Asuransi*

iv. *Endpoint* /revenue/export/status

Tujuan utama dari *endpoint API* ini adalah untuk mengetahui status proses ekspor yang sedang dilakukan, *endpoint* ini dimaksudkan untuk di *hit* berulang-ulang oleh tim *Front End* untuk mengetahui *progress* proses ekspor dengan bentuk persentase. Jika proses ekspor telah selesai, maka *endpoint* akan mengembalikan *link url* untuk mengunduh *file excel* hasil ekspor tersebut. Secara sederhana, alur jalannya program yang dibuat untuk melakukan ekspor data *revenue* Reksa Dana dan asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* pada Gambar 3.10




Gambar 3.10. *Flowchart API /revenue/export/status* di Modul Reksa Dana dan Asuransi

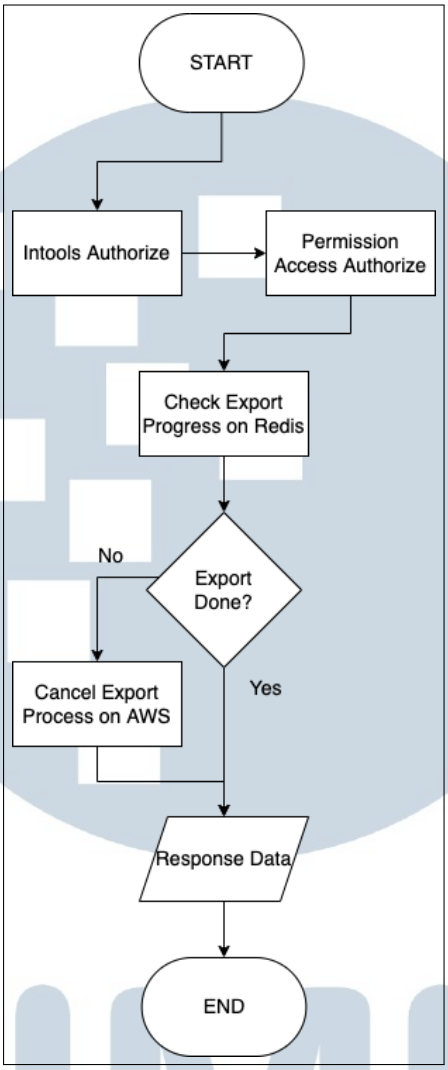
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.11.

```
type Response struct {
    Data struct {
        PresignedURL string `json:"presigned_url"`
        Status       string `json:"status"`
        Caption      string `json:"caption"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.11. *Struktur data response API /revenue/export/status di Modul Reksa Dana dan Asuransi*

v. *Endpoint* /revenue/export/cancel

Tujuan utama dari *endpoint API* ini adalah untuk melakukan pembatalan proses ekspor data yang sedang berlangsung. Karena proses ekspor data yang dilakukan adalah dalam jumlah data yang besar dan menyebabkan proses ekspor data membutuhkan waktu yang cukup lama, maka ada kemungkinan bahwa pengguna membatalkan proses ekspor tersebut baik secara langsung maupun tidak langsung. Secara sederhana, alur jalannya program yang dibuat untuk melakukan pembatalan ekspor data *revenue* Reksa Dana dan asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* pada Gambar 3.12




Gambar 3.12. *Flowchart API /revenue/export/cancel* di Modul Reksa Dana dan Asuransi

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.13.

```
type Response struct {
    Data struct {
        PresignedURL string `json:"presigned_url"`
        Status       string `json:"status"`
        Caption      string `json:"caption"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.13. *Struktur data response API /revenue/export/cancel di Modul Reksa Dana dan Asuransi*

2. Modul Kamus Asuransi Tokopedia

(a) Persyaratan dan Perancangan

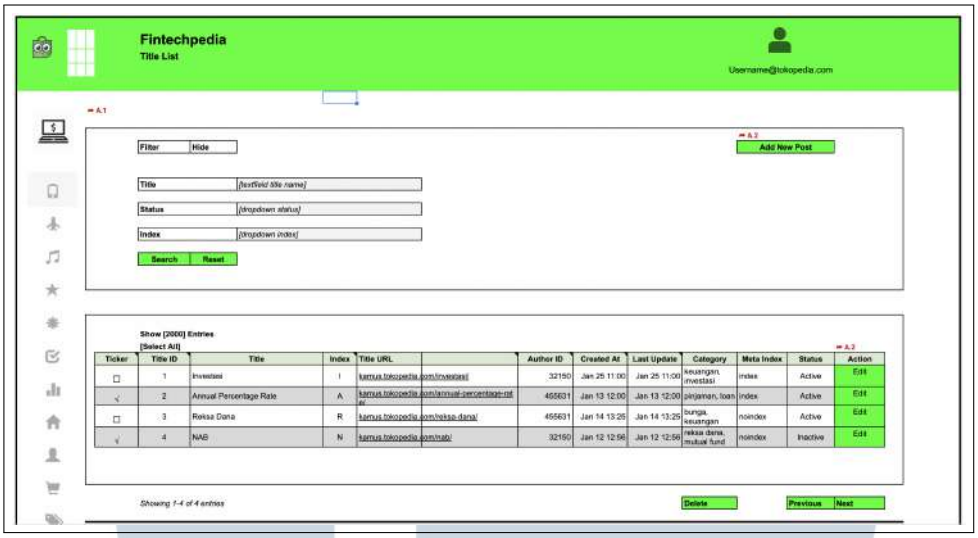
Dalam pengembangan modul kamus asuransi ini, bagian yang ingin difokuskan adalah halaman manajemen *post*. Terdapat beberapa persyaratan dalam pengembangan halaman ini diantaranya:

- Menggunakan bahasa pemrograman Go
- Menggunakan basis data MySQL
- Menggunakan sistem caching Redis
- Menggunakan metode penulisan *Clean Code*
- Membuat *Unit Test* untuk setiap baris kode

Terdapat beberapa fitur yang diharapkan pada halaman manajemen pendapatan, diantaranya:

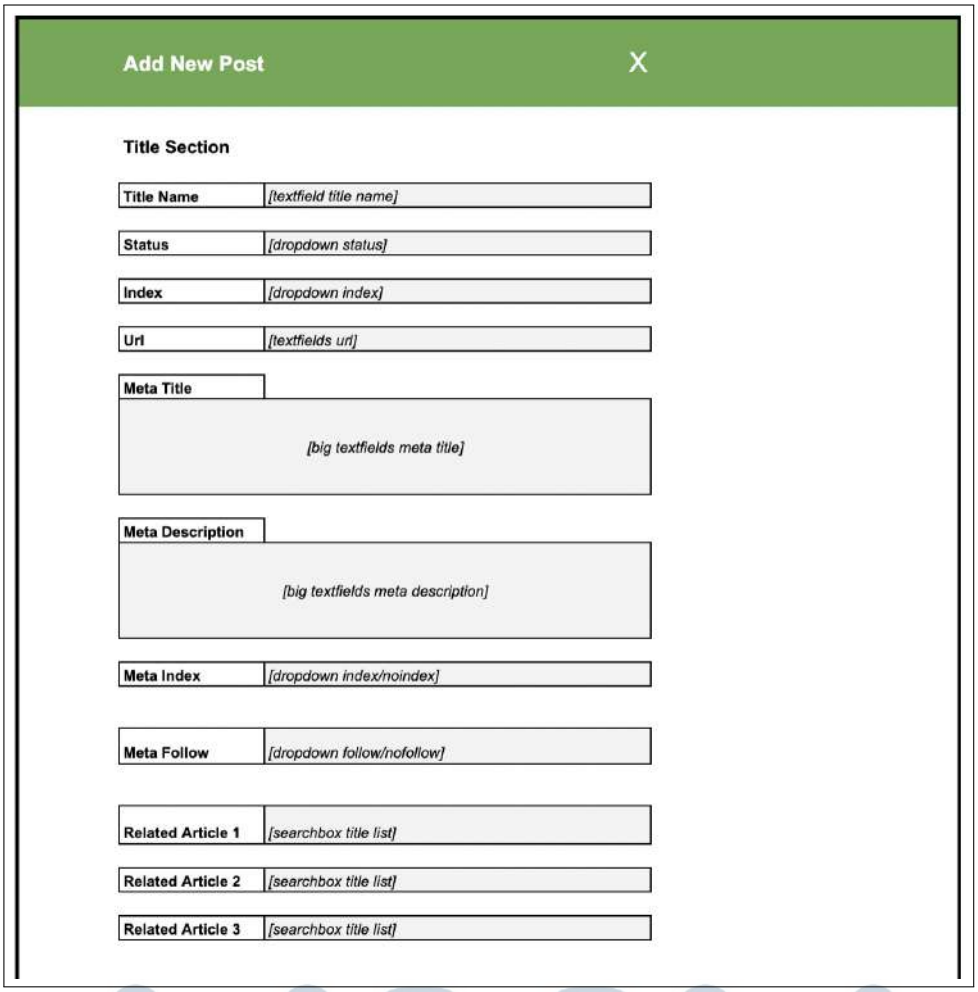
- Menampilkan semua data *post* dengan fitur *limit, paging, sort, dan filter*
- Menampilkan detail data *post*
- Menampilkan *filter select option* berdasarkan kategori, CTA, status, index, meta index, dan meta follow
- Menambah data *post* lengkap dengan *multiple* kategori, CTA, konten, dan definisinya
- Mengubah data *post* lengkap dengan *multiple* kategori, CTA, konten, dan definisinya

Selain persyaratan di atas, terdapat tahap perancangan, telah disediakan *mock up* gambaran halaman terkait halaman manajemen post seperti terlampir pada Gambar 3.14



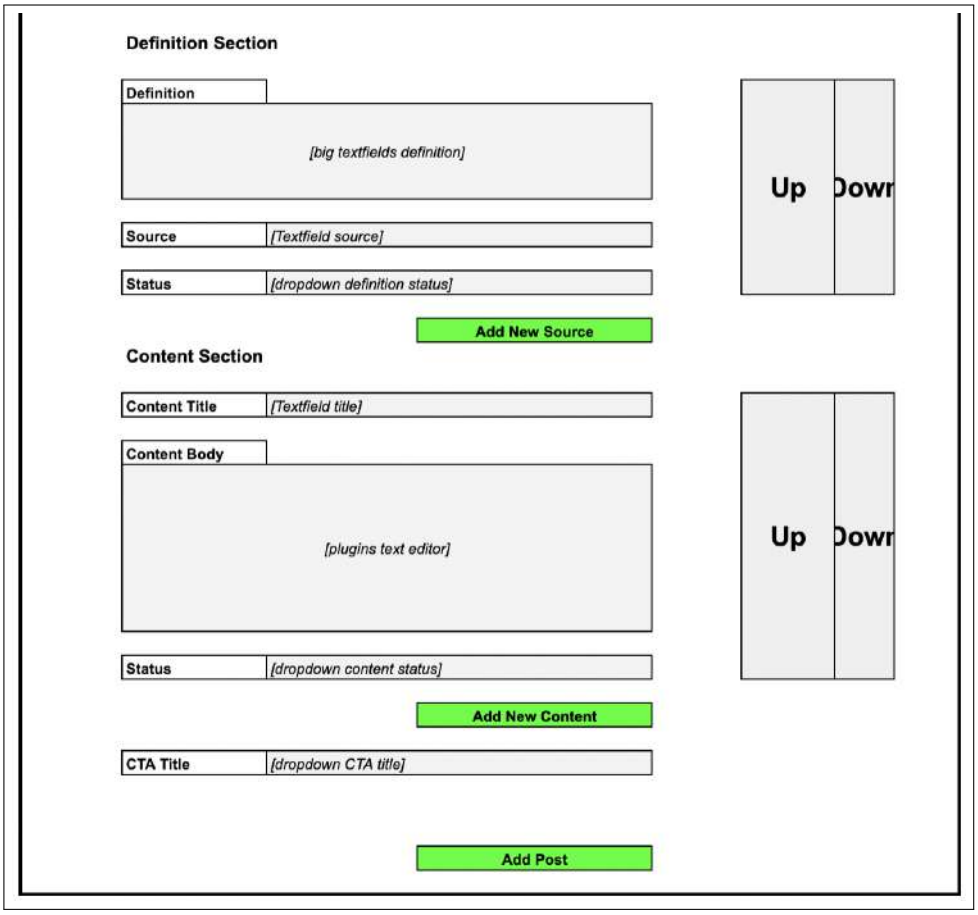

Gambar 3.14. Mock Up Modul Kamus Asuransi Halaman Manajemen *Post*

Selain itu juga terdapat *mock up* untuk fitur menambah dan mengubah *post* seperti terlampir pada Gambar 3.15




Gambar 3.15. Mock Up Modul Kamus Asuransi Fitur Tambah dan Ubah *Post*

Pada fitur menambah dan mengubah *post* juga terdapat *field* untuk mengisi definisi, konten, dan CTA yang sifatnya *multiple*, dimana dapat memasukkan lebih dari 1 *item* seperti terlampir pada Gambar 3.16.




Gambar 3.16. Mock Up Modul Kamus Asuransi Fitur Tambah dan Ubah *Post* Lanjutan

Pada Gambar 3.14, 3.15, dan 3.16 telah ditampilkan gambaran dari halaman yang nantinya akan diimplementasikan oleh tim *front end*. Dari *mock up* tersebut kemudian didefinisikan *API endpoint* yang akan dikembangkan, diantaranya:

i. *Endpoint* /post
   Bertujuan untuk menampilkan semua data *post* dengan fitur *limit, paging, sort, dan filter*
ii. *Endpoint* /post/detail/:id
   Bertujuan untuk menampilkan informasi mendetail dari *post* berdasarkan id *post* tersebut
iii. *Endpoint* /post/filter/category
   Bertujuan untuk menampilkan data category yang dapat digunakan untuk filter dalam menampilkan data *post*
iv. *Endpoint* /post/filter/cta
   Bertujuan untuk menampilkan data cta yang dapat digunakan untuk

filter dalam menampilkan data *post*

v. *Endpoint* /post/filter/status
Bertujuan untuk menampilkan data status yang dapat digunakan untuk filter dalam menampilkan data *post*

vi. *Endpoint* /post/filter/index
Bertujuan untuk menampilkan data index yang dapat digunakan untuk filter dalam menampilkan data *post*

vii. *Endpoint* /post/filter/meta-index
Bertujuan untuk menampilkan data meta index yang dapat digunakan untuk filter dalam menampilkan data *post*

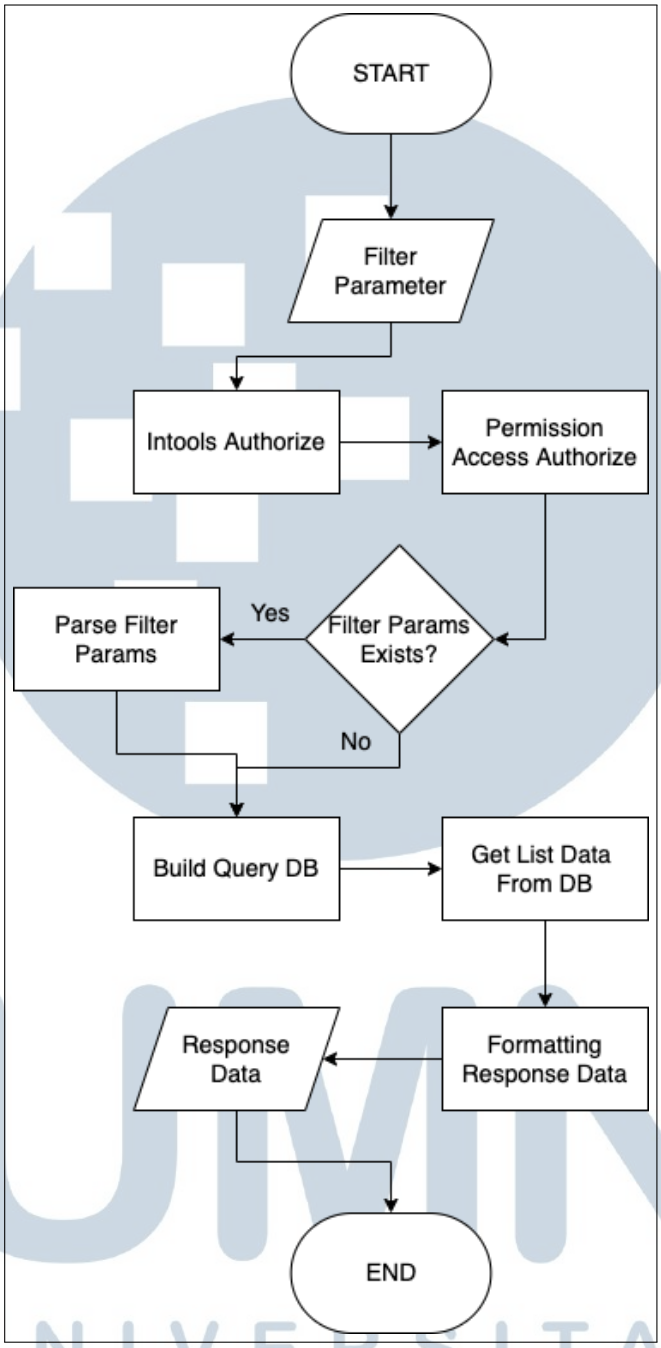
viii. *Endpoint* /post/filter/meta-follow
Bertujuan untuk menampilkan data meta follow yang dapat digunakan untuk filter dalam menampilkan data *post*

ix. *Endpoint* /post/create
Bertujuan untuk membuat *post* baru dengan mengirimkan data json sesuai format ketentuan data

x. *Endpoint* /post/update/:id
Bertujuan untuk mengubah informasi data *post* dengan mengirimkan data json sesuai format ketentuan data

(b) Implementasi

i. *Endpoint* /post
Tujuan utama dari *endpoint API* ini adalah untuk mendapatkan daftar data *post* kamus di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk mendapatkan data *post* kamus di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.17




Gambar 3.17. *Flowchart API /post* di Modul Kamus Asuransi Tokopedia

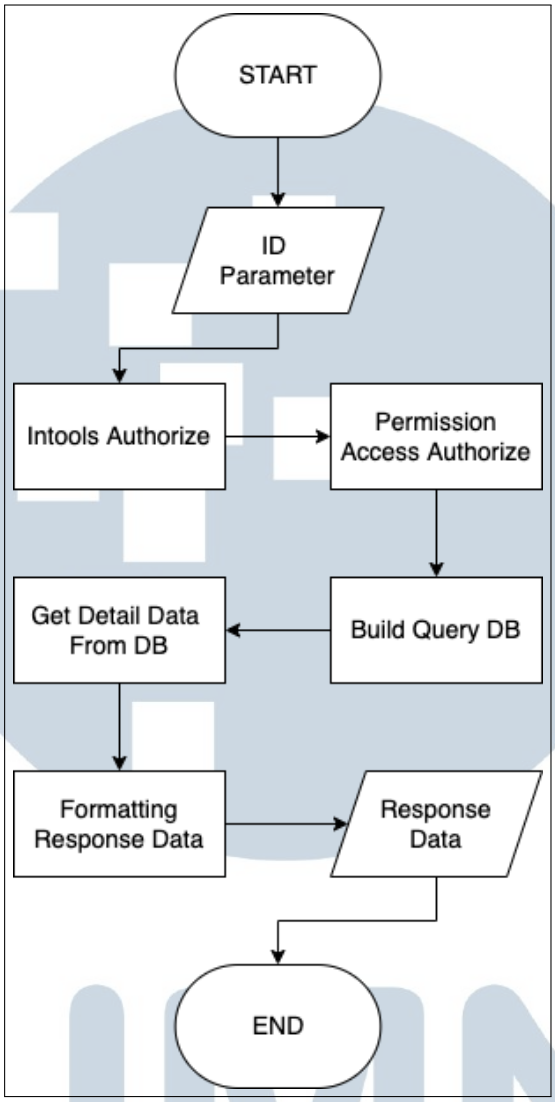
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.18

```
type Response struct {
    Data struct {
        Table struct {
            Total     int `json:"total"`
            TotalPage int `json:"total_page"`
            Page      int `json:"page"`
            Limit     int `json:"limit"`
            Rows      []struct {
                ID              int       `json:"id"`
                Title           string    `json:"title"`
                LetterIndex     string    `json:"letter_index"`
                Status          int       `json:"status"`
                Category        string    `json:"category"`
                PreferenceURL   string    `json:"preference_url"`
                MetaTitle       string    `json:"meta_title"`
                MetaDescription string    `json:"meta_description"`
                IsIndex         string    `json:"is_index"`
                IsFollow        string    `json:"is_follow"`
                IsPopular       string    `json:"is_popular"`
                TkpUserIDCreate int       `json:"tkp_user_id_create"`
                CreatedAt       time.Time `json:"created_at"`
                TkpUserIDUpdate int       `json:"tkp_user_id_update"`
                UpdatedAt       time.Time `json:"updated_at"`
                RelatedPost1    int       `json:"related_post_1"`
                RelatedPost2    int       `json:"related_post_2"`
                RelatedPost3    int       `json:"related_post_3"`
            } `json:"rows"`
        } `json:"table"`
        Cards interface{} `json:"cards"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.18. *Struktur data response API /post di Modul Kamus Asuransi Tokopedia*

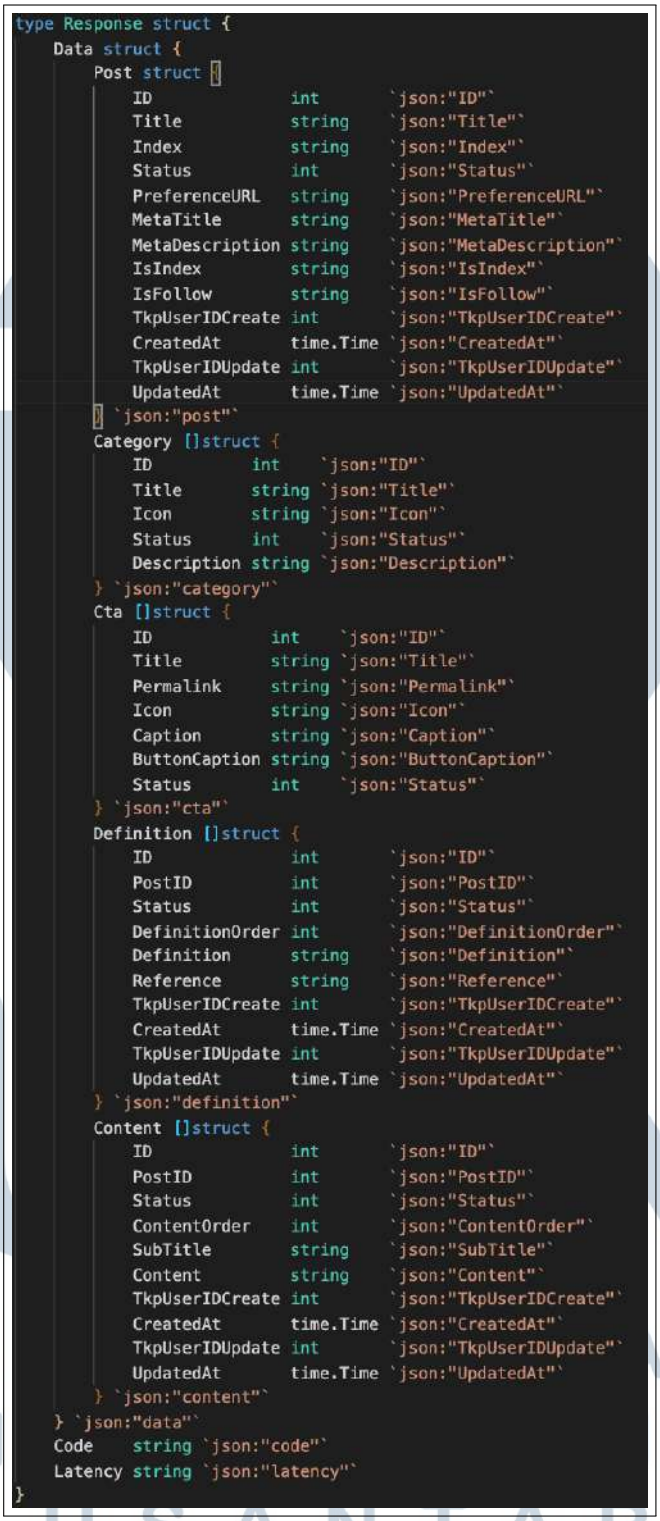
ii. *Endpoint* /post/detail/:id

Tujuan utama dari *endpoint API* ini adalah untuk mendapatkan detail data *post* kamus di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk mendapatkan data *post* kamus di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.19




Gambar 3.19. *Flowchart API /post/detail* di Modul Kamus Asuransi Tokopedia

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.20



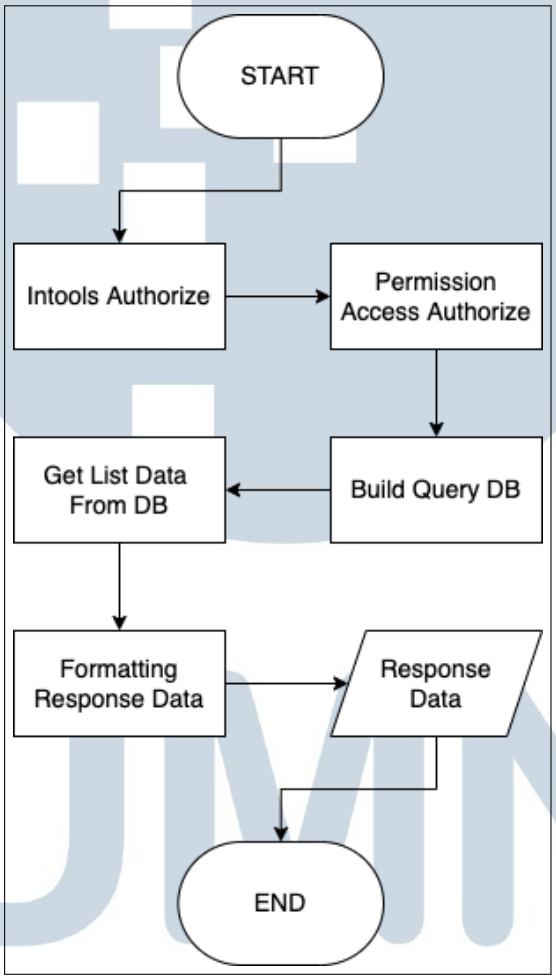
```
type Response struct {
    Data struct {
        Post struct {
            ID              int       `json:"ID"`
            Title           string    `json:"Title"`
            Index           string    `json:"Index"`
            Status          int       `json:"Status"`
            PreferenceURL   string    `json:"PreferenceURL"`
            MetaTitle       string    `json:"MetaTitle"`
            MetaDescription string    `json:"MetaDescription"`
            IsIndex         string    `json:"IsIndex"`
            IsFollow        string    `json:"IsFollow"`
            TkpUserIDCreate int       `json:"TkpUserIDCreate"`
            CreatedAt       time.Time `json:"CreatedAt"`
            TkpUserIDUpdate int       `json:"TkpUserIDUpdate"`
            UpdatedAt       time.Time `json:"UpdatedAt"`
        } `json:"post"`
        Category []struct {
            ID          int    `json:"ID"`
            Title       string `json:"Title"`
            Icon        string `json:"Icon"`
            Status      int    `json:"Status"`
            Description string `json:"Description"`
        } `json:"category"`
        Cta []struct {
            ID            int    `json:"ID"`
            Title         string `json:"Title"`
            Permalink     string `json:"Permalink"`
            Icon          string `json:"Icon"`
            Caption       string `json:"Caption"`
            ButtonCaption string `json:"ButtonCaption"`
            Status        int    `json:"Status"`
        } `json:"cta"`
        Definition []struct {
            ID              int       `json:"ID"`
            PostID          int       `json:"PostID"`
            Status          int       `json:"Status"`
            DefinitionOrder int       `json:"DefinitionOrder"`
            Definition      string    `json:"Definition"`
            Reference       string    `json:"Reference"`
            TkpUserIDCreate int       `json:"TkpUserIDCreate"`
            CreatedAt       time.Time `json:"CreatedAt"`
            TkpUserIDUpdate int       `json:"TkpUserIDUpdate"`
            UpdatedAt       time.Time `json:"UpdatedAt"`
        } `json:"definition"`
        Content []struct {
            ID              int       `json:"ID"`
            PostID          int       `json:"PostID"`
            Status          int       `json:"Status"`
            ContentOrder    int       `json:"ContentOrder"`
            SubTitle        string    `json:"SubTitle"`
            Content         string    `json:"Content"`
            TkpUserIDCreate int       `json:"TkpUserIDCreate"`
            CreatedAt       time.Time `json:"CreatedAt"`
            TkpUserIDUpdate int       `json:"TkpUserIDUpdate"`
            UpdatedAt       time.Time `json:"UpdatedAt"`
        } `json:"content"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.20. *Struktur data response API /post/detail di Modul Kamus Asuransi Tokopedia*

iii. *Endpoint* /post/filter

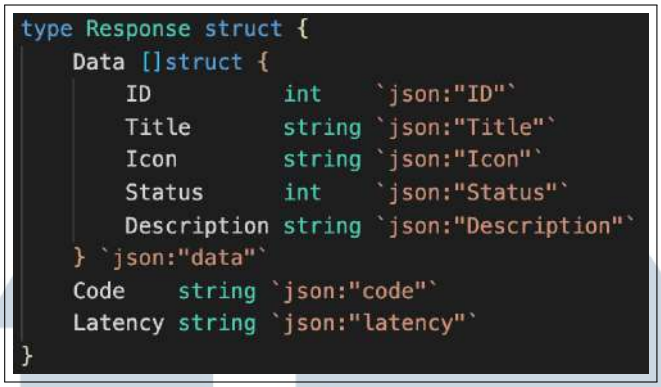
Tujuan utama dari *endpoint API* ini adalah untuk mendapatkan

daftar data filter untuk menyaring daftar data *post* di kamus Asuransi Tokopedia. Terdapat beberapa *sub-endpoint* filter yaitu filter berdasarkan *category*, *cta*, *status*, *index*, *meta-index*, dan *meta-follow*. Secara sederhana, alur jalannya program yang dibuat untuk mendapatkan data filter ini didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.21




Gambar 3.21. *Flowchart API /post/filter* di Modul Kamus Asuransi Tokopedia

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.22

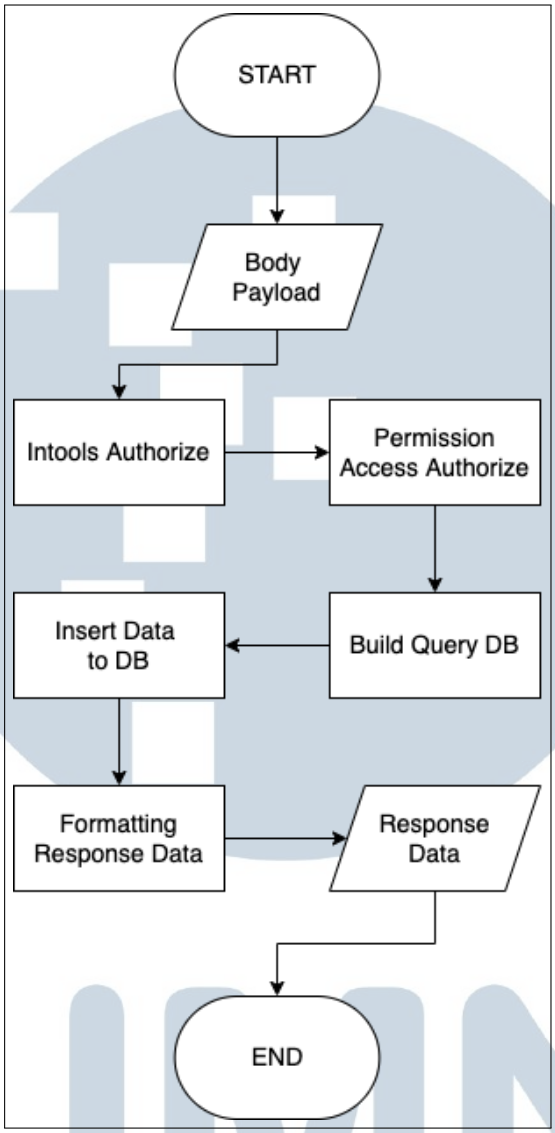


```
type Response struct {
    Data []struct {
        ID          int    `json:"ID"`
        Title       string `json:"Title"`
        Icon        string `json:"Icon"`
        Status      int    `json:"Status"`
        Description string `json:"Description"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.22. *Struktur data response API /post/filter di Modul Kamus Asuransi Tokopedia*

iv. *Endpoint* /post/create

Tujuan utama dari *endpoint API* ini adalah untuk membuat data *post* kamus baru di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk membuat data *post* kamus di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.23




Gambar 3.23. *Flowchart API /post/create* di Modul Kamus Asuransi Tokopedia

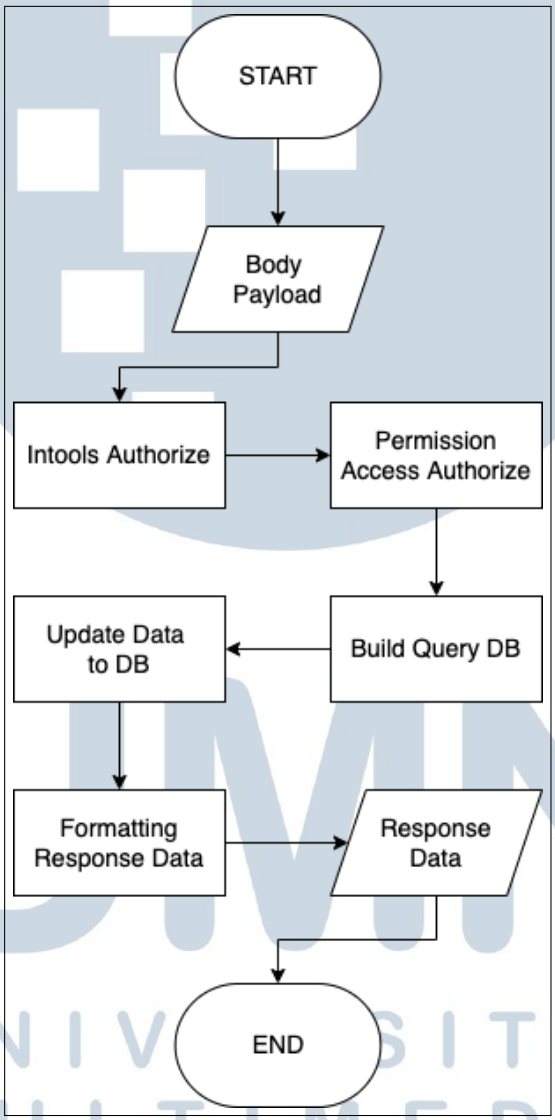
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.24

```
type Response struct {
    Data struct {
        Success bool   `json:"Success"`
        Message string `json:"Message"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.24. *Struktur data response API /post/create di Modul Kamus Asuransi Tokopedia*

v. *Endpoint* /post/update/:id

Tujuan utama dari *endpoint API* ini adalah untuk mengubah data *post* kamus di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk mendapatkan data *post* kamus di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.25




Gambar 3.25. *Flowchart API /post/update* di Modul Kamus Asuransi Tokopedia

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.26


Gambar 3.26. *Struktur data response API /post/update di Modul Kamus Asuransi Tokopedia*

3. Modul Asuransi Penerbangan

(a) Persyaratan dan Perancangan

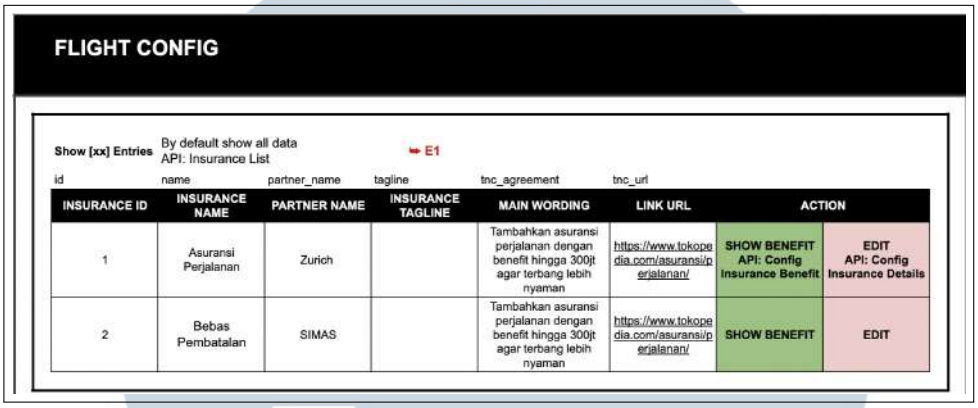
Dalam pengembangan modul kamus asuransi ini, bagian yang ingin difokuskan adalah halaman manajemen konfigurasi master data asuransi. Terdapat beberapa persyaratan dalam pengembangan halaman ini diantaranya:

- Menggunakan bahasa pemrograman Go
- Menggunakan basis data MySQL
- Menggunakan sistem caching Redis
- Menggunakan metode penulisan *Clean Code*
- Membuat *Unit Test* untuk setiap baris kode

Terdapat beberapa fitur yang diharapkan pada halaman manajemen pendapatan, diantaranya:

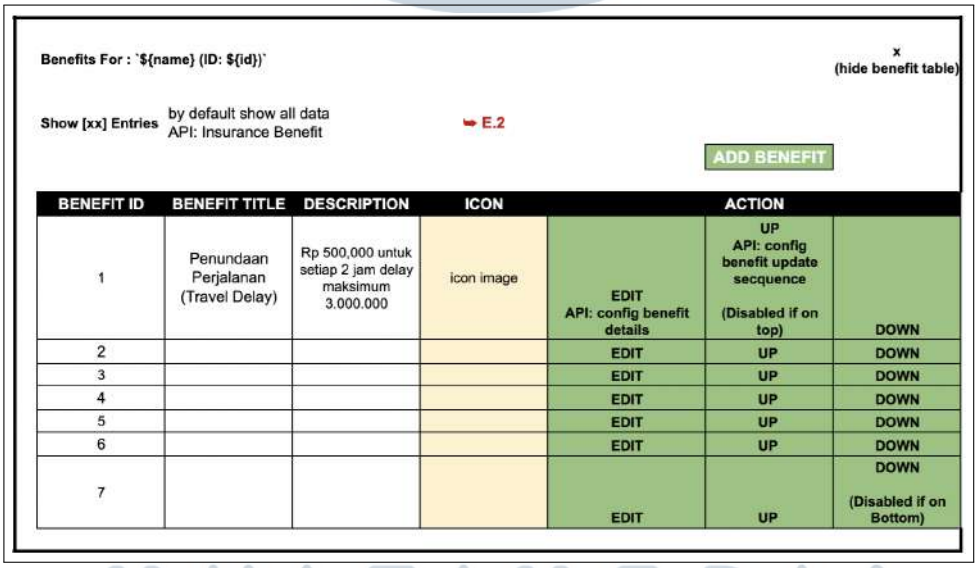
- Menampilkan semua data master asuransi dengan fitur *limit, paging, sort, dan filter*
- Menampilkan detail data master asuransi
- Mengubah data master asuransi
- Menampilkan daftar keuntungan / *benefit* dari asuransi yang dipilih
- Menambah data keuntungan / *benefit* dari asuransi yang dipilih
- Mengubah data keuntungan / *benefit* asuransi
- Mengubah urutan *sequence* data keuntungan / *benefit* asuransi
- Menampilkan *filter select option* berdasarkan kategori, CTA, status, index, meta index, dan meta follow

Selain persyaratan di atas, terdapat tahap perancangan, telah disediakan *mock up* gambaran halaman terkait halaman manajemen konfigurasi asuransi seperti terlampir pada Gambar 3.27.



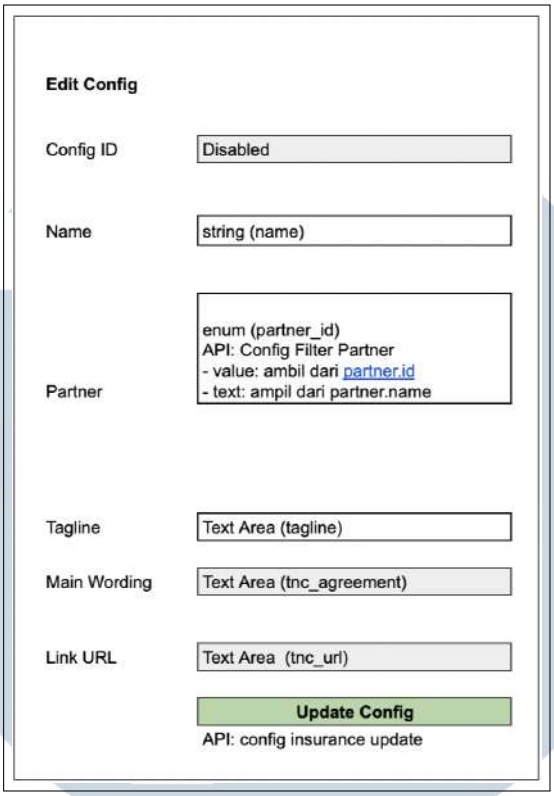

Gambar 3.27. Mock Up Modul Asuransi Penerbangan Halaman Manajemen Konfigurasi

Selain itu, pada Gambar 3.28 adalah gambaran ketika pengguna melihat dan mengubah *sequence* urutan daftar keuntungan / *benefit* dari asuransi yang dipilih.



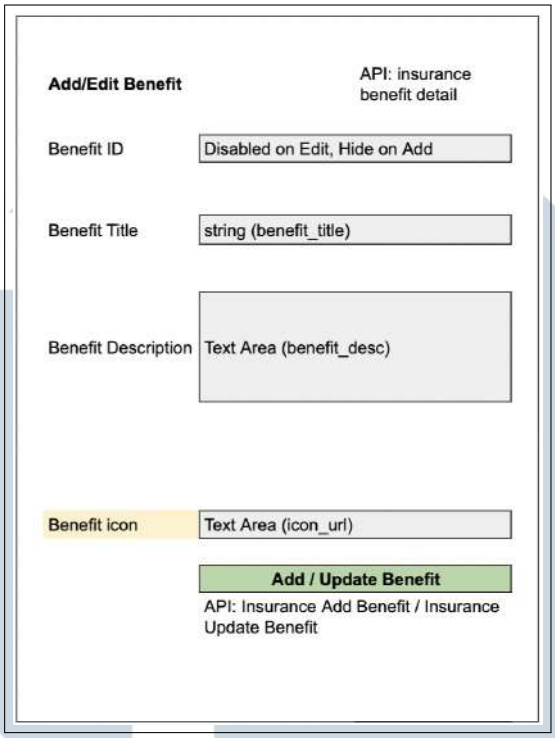

Gambar 3.28. Mock Up Modul Asuransi Penerbangan Halaman Detail Konfigurasi Keuntungan Asuransi

Kemudian, pada Gambar 3.29 adalah gambaran ketika pengguna melihat dan mengubah detail data asuransi dari asuransi yang dipilih.




Gambar 3.29. Mock Up Modul Asuransi Penerbangan Modal Detail Konfigurasi Asuransi

Kemudian, pada Gambar 3.30 adalah gambaran ketika pengguna melihat dan mengubah detail data keuntungan dari asuransi yang dipilih.




Gambar 3.30. Mock Up Modul Asuransi Penerbangan Modal Detail Konfigurasi Keuntungan Asuransi

Pada Gambar 3.27, 3.28, 3.29 dan 3.30 telah ditampilkan gambaran dari halaman yang nantinya akan diimplementasikan oleh tim *front end*. Dari *mock up* tersebut kemudian didefinisikan *API endpoint* yang akan dikembangkan, diantaranya:

i. *Endpoint* /configuration/insurance
   Bertujuan untuk menampilkan semua data konfigurasi asuransi dengan fitur *limit, paging, sort, dan filter*
ii. *Endpoint* /configuration/insurance/:id/detail
   Bertujuan untuk menampilkan informasi mendetail dari konfigurasi asuransi berdasarkan id asuransi yang dipilih.
iii. *Endpoint* /configuration/insurance/:id/update
   Bertujuan untuk mengubah data informasi dari konfigurasi asuransi berdasarkan id asuransi yang dipilih.
iv. *Endpoint* /configuration/insurance/:id/benefit
   Bertujuan untuk menampilkan informasi daftar keuntungan berdasarkan id asuransi yang dipilih.
v. *Endpoint* /configuration/insurance/:id/benefit/add

Bertujuan untuk menambahkan data keuntungan berdasarkan id asuransi yang dipilih.

vi. *Endpoint* /configuration/benefit/detail/:id
Bertujuan untuk menampilkan informasi mendetail dari keuntungan asuransi berdasarkan id keuntungan yang dipilih.

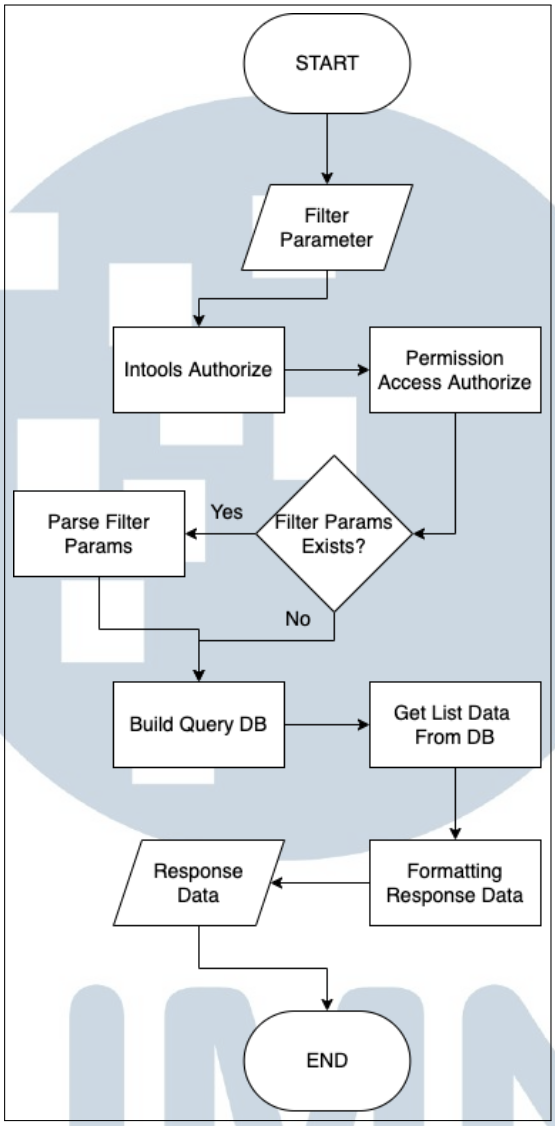
vii. *Endpoint* /configuration/benefit/update/:id
Bertujuan untuk mengubah data informasi dari keuntungan asuransi berdasarkan id keuntungan yang dipilih.

viii. *Endpoint* /configuration/benefit/update-sequence/:id
Bertujuan untuk mengubah urutan *sequence* data keuntungan asuransi berdasarkan id asuransi yang dipilih.

ix. *Endpoint* /configuration/filter/partner
Bertujuan untuk menampilkan data partner yang dapat digunakan untuk filter dalam menampilkan data asuransi

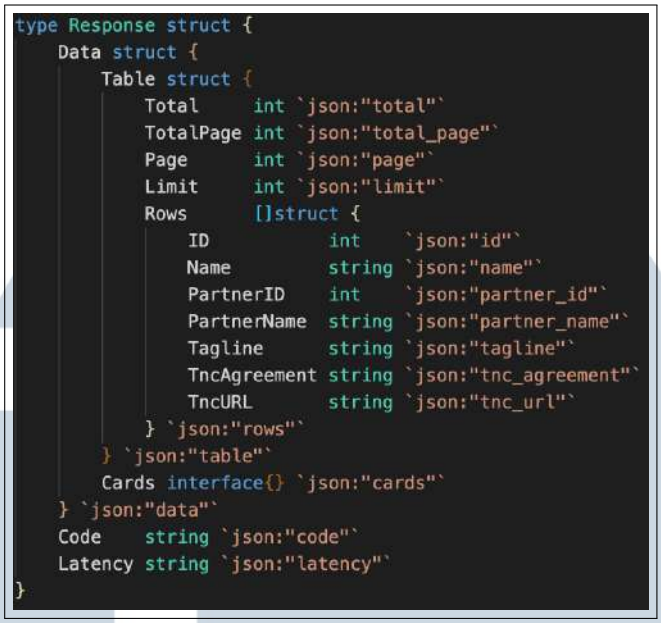
(b) Implementasi

i. *Endpoint* /configuration/insurance
Tujuan utama dari *endpoint API* ini adalah untuk mendapatkan daftar data master konfigurasi asuransi di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk mendapatkan data master asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.31




Gambar 3.31. *Flowchart API /configuration/insurance* di Modul Asuransi Penerbangan

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.32

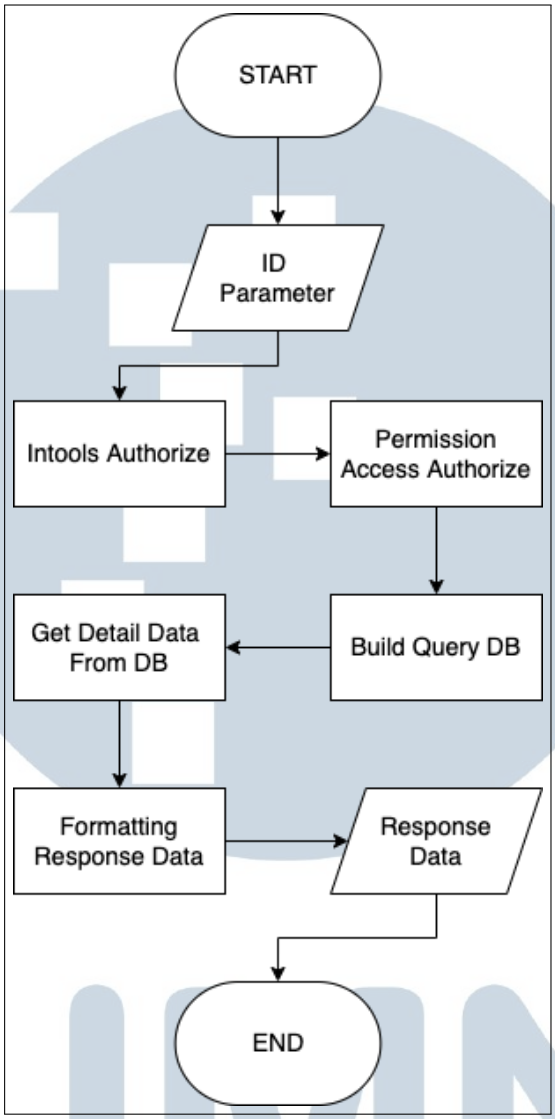


```
type Response struct {
    Data struct {
        Table struct {
            Total     int `json:"total"`
            TotalPage int `json:"total_page"`
            Page      int `json:"page"`
            Limit     int `json:"limit"`
            Rows      []struct {
                ID          int    `json:"id"`
                Name        string `json:"name"`
                PartnerID   int    `json:"partner_id"`
                PartnerName string `json:"partner_name"`
                Tagline     string `json:"tagline"`
                TncAgreement string `json:"tnc_agreement"`
                TncURL      string `json:"tnc_url"`
            } `json:"rows"`
        } `json:"table"`
        Cards interface{} `json:"cards"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.32. *Struktur data response API /configuration/insurance di Modul Asuransi Penerbangan*

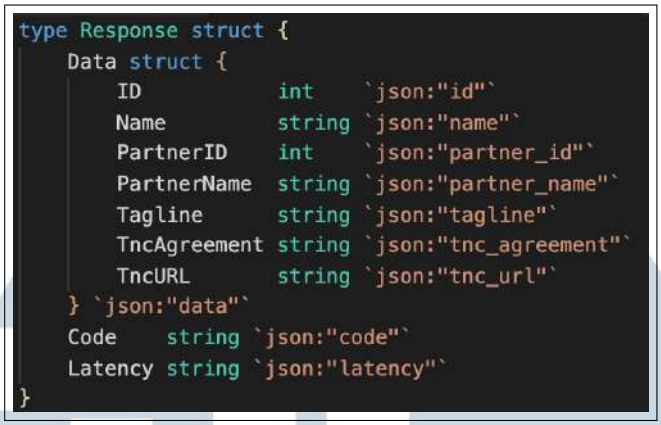
ii. *Endpoint* /configuration/insurance/:id/detail

Tujuan utama dari *endpoint API* ini adalah untuk mendapatkan detail data master konfigurasi asuransi di Tokopedia berdasarkan id asuransi. Secara sederhana, alur jalannya program yang dibuat untuk mendapatkan detail data master asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.33




Gambar 3.33. *Flowchart API /configuration/insurance/:id/detail* di Modul Asuransi Penerbangan

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.34

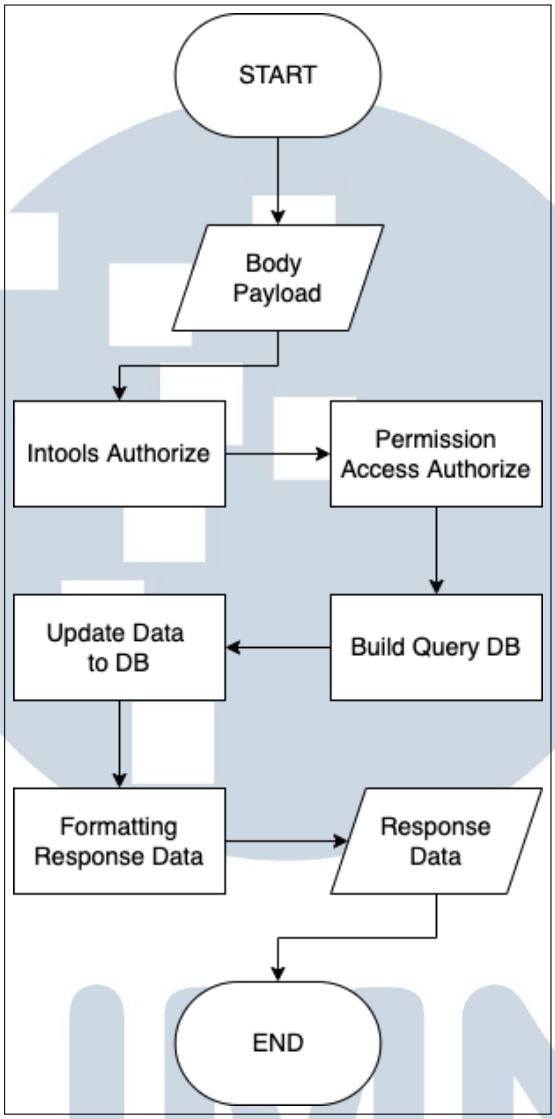


```
type Response struct {
    Data struct {
        ID           int    `json:"id"`
        Name         string `json:"name"`
        PartnerID    int    `json:"partner_id"`
        PartnerName  string `json:"partner_name"`
        Tagline      string `json:"tagline"`
        TncAgreement string `json:"tnc_agreement"`
        TncURL       string `json:"tnc_url"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.34. *Struktur data response API /configuration/insurance/detail di Modul Asuransi Penerbangan*

iii. *Endpoint* /configuration/insurance/:id/update

Tujuan utama dari *endpoint API* ini adalah untuk mengubah data master konfigurasi asuransi di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk mengubah data master asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.35




Gambar 3.35. *Flowchart API /configuration/insurance/:id/update* di Modul Asuransi Penerbangan

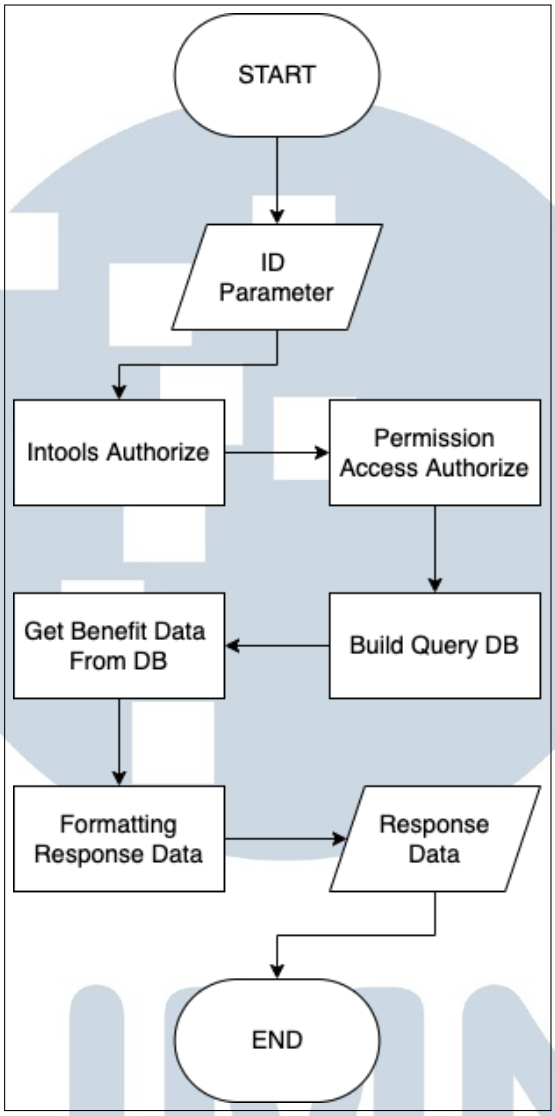
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.36

```
type Response struct {
    Data struct {
        Success bool   `json:"Success"`
        Message string `json:"Message"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.36. *Struktur data response API /configuration/insurance/:id/update di Modul Asuransi Penerbangan*

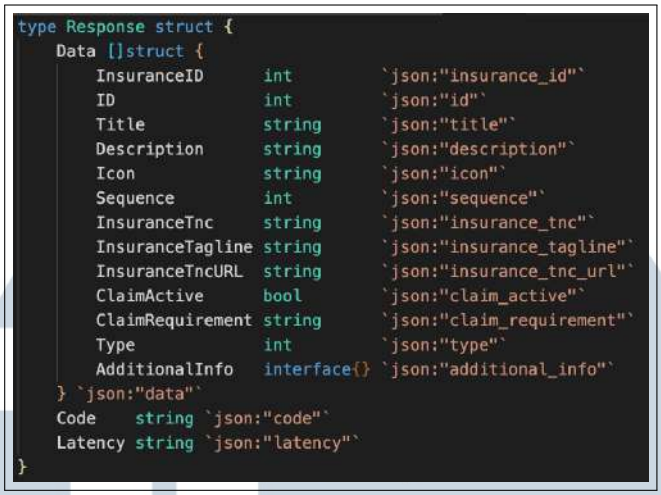
iv. *Endpoint* /configuration/insurance/:id/benefit

Tujuan utama dari *endpoint API* ini adalah untuk menampilkan daftar data keuntungan / *benefit* dari asuransi di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk menampilkan daftar data keuntungan asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.37




Gambar 3.37. *Flowchart API /configuration/insurance/:id/benefit* di Modul Asuransi Penerbangan

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.38

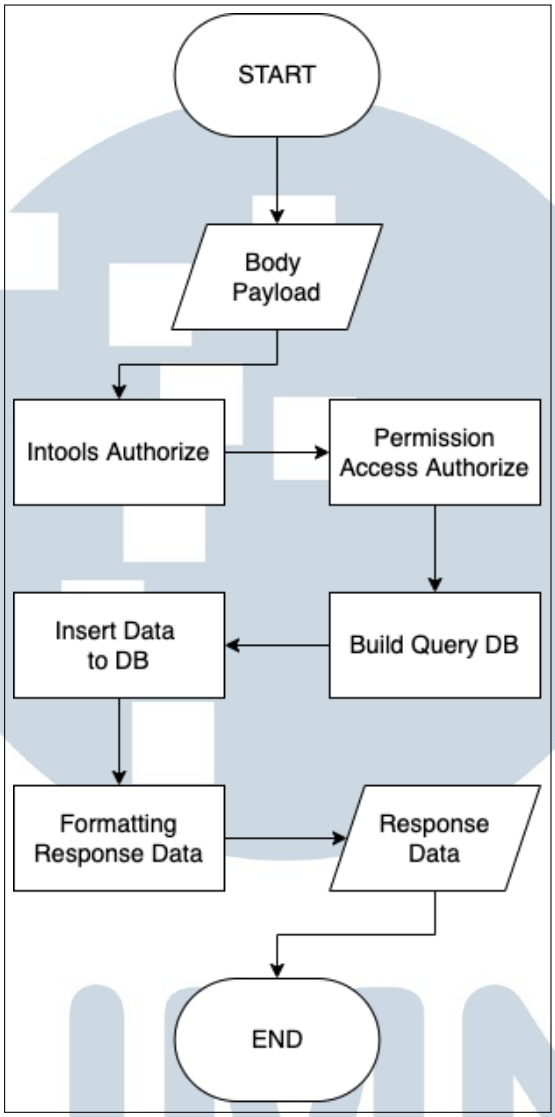


```
type Response struct {
    Data []struct {
        InsuranceID       int         `json:"insurance_id"`
        ID                int         `json:"id"`
        Title             string      `json:"title"`
        Description       string      `json:"description"`
        Icon              string      `json:"icon"`
        Sequence          int         `json:"sequence"`
        InsuranceTnc      string      `json:"insurance_tnc"`
        InsuranceTagline  string      `json:"insurance_tagline"`
        InsuranceTncURL   string      `json:"insurance_tnc_url"`
        ClaimActive       bool        `json:"claim_active"`
        ClaimRequirement  string      `json:"claim_requirement"`
        Type              int         `json:"type"`
        AdditionalInfo    interface{} `json:"additional_info"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.38. *Struktur data response API /configuration/insurance/:id/benefit di Modul Asuransi Penerbangan*

v. *Endpoint* /configuration/insurance/:id/benefit/add

Tujuan utama dari *endpoint API* ini adalah untuk menambahkan data keuntungan / *benefit* baru dari asuransi yang dipilih. Secara sederhana, alur jalannya program yang dibuat untuk menambahkan data keuntungan asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.39




Gambar 3.39. *Flowchart API /configuration/insurance/:id/benefit/add* di Modul Asuransi Penerbangan

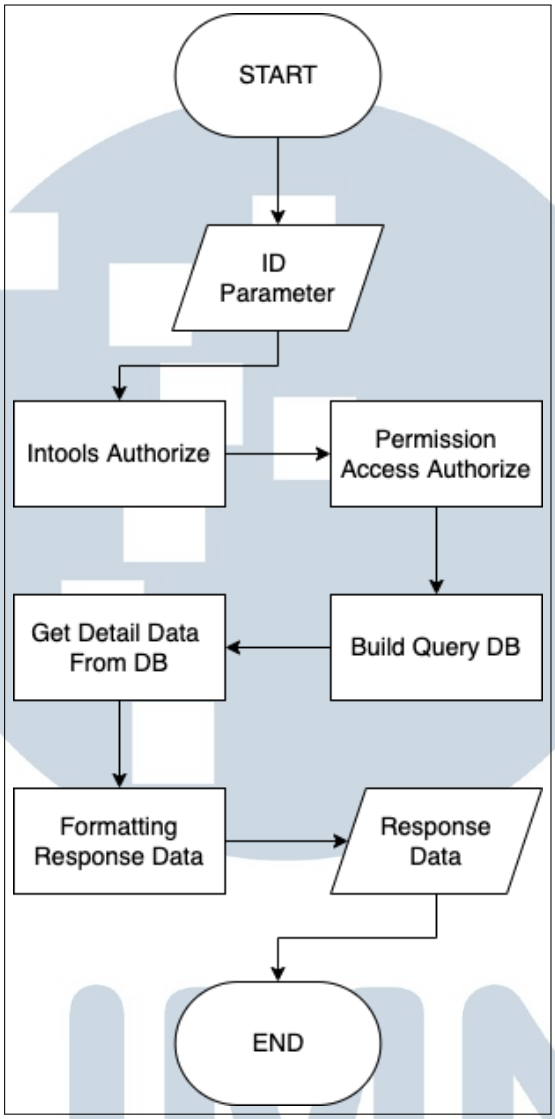
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.40

```
type Response struct {
    Data struct {
        Success bool   `json:"Success"`
        Message string `json:"Message"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.40. *Struktur data response API /configuration/insurance/:id/benefit/add di Modul Asuransi Penerbangan*

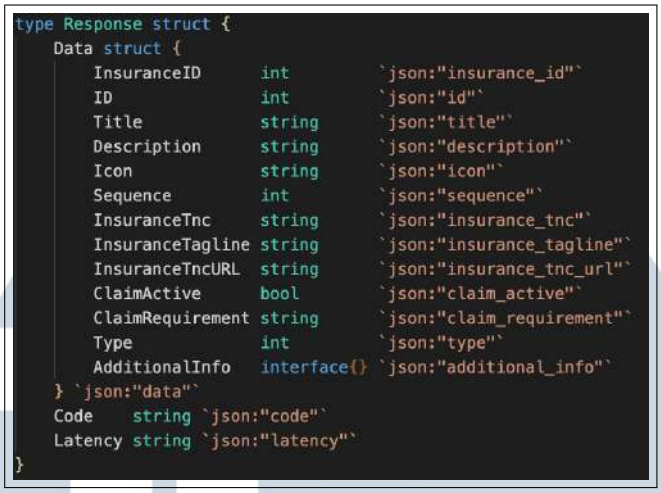
vi. *Endpoint* /configuration/benefit/detail/:id

Tujuan utama dari *endpoint API* ini adalah untuk menampilkan detail data keuntungan / *benefit* dari asuransi yang dipilih. Secara sederhana, alur jalannya program yang dibuat untuk menampilkan detail data keuntungan asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.41




Gambar 3.41. *Flowchart API /configuration/benefit/detail/:id* di Modul Asuransi Penerbangan

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.42

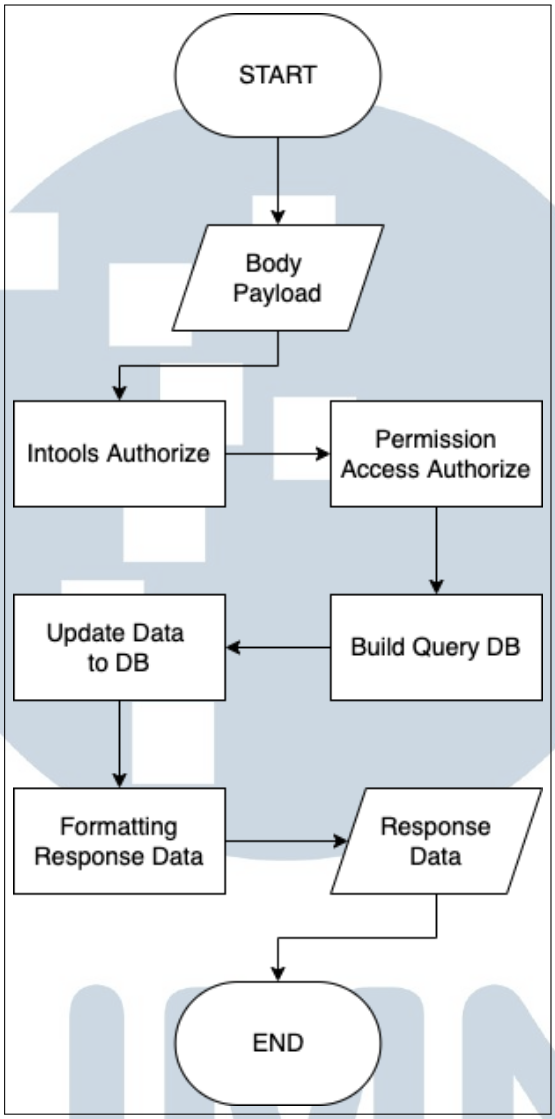


```
type Response struct {
    Data struct {
        InsuranceID      int         `json:"insurance_id"`
        ID               int         `json:"id"`
        Title            string      `json:"title"`
        Description      string      `json:"description"`
        Icon             string      `json:"icon"`
        Sequence         int         `json:"sequence"`
        InsuranceTnc     string      `json:"insurance_tnc"`
        InsuranceTagline string      `json:"insurance_tagline"`
        InsuranceTncURL  string      `json:"insurance_tnc_url"`
        ClaimActive      bool        `json:"claim_active"`
        ClaimRequirement string      `json:"claim_requirement"`
        Type             int         `json:"type"`
        AdditionalInfo   interface{} `json:"additional_info"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.42. *Struktur data response API /configuration/benefit/detail/:id di Modul Asuransi Penerbangan*

vii. *Endpoint* /configuration/benefit/update/:id

Tujuan utama dari *endpoint API* ini adalah untuk mengubah data keuntungan / *benefit* dari asuransi yang dipilih. Secara sederhana, alur jalannya program yang dibuat untuk mengubah detail data keuntungan asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.43




Gambar 3.43. *Flowchart API /configuration/benefit/update/:id* di Modul Asuransi Penerbangan

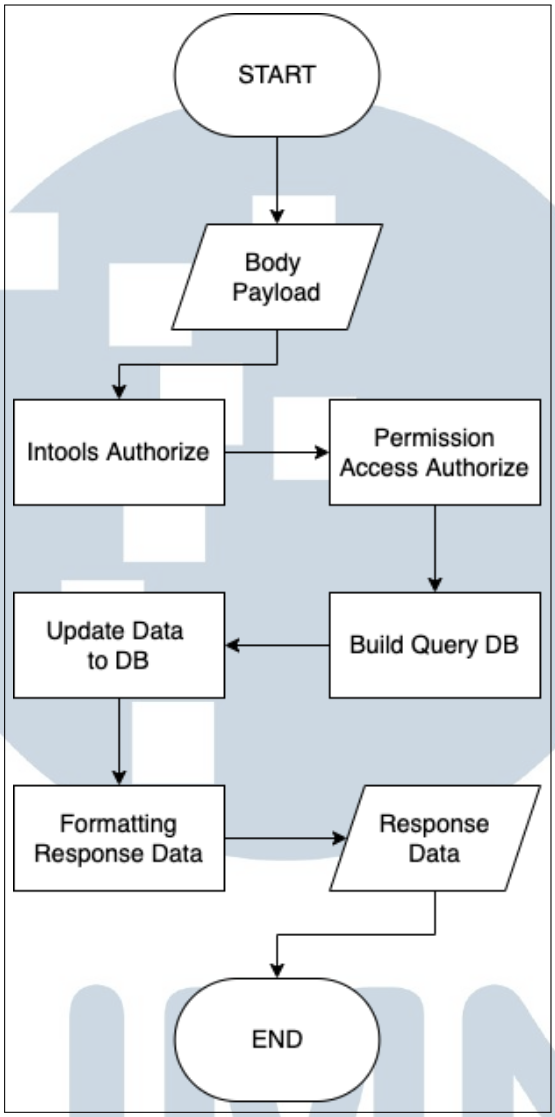
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.44

```
type Response struct {
    Data struct {
        Success bool   `json:"Success"`
        Message string `json:"Message"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.44. *Struktur data response API /configuration/benefit/update/:id di Modul Asuransi Penerbangan*

viii. *Endpoint* /configuration/benefit/update-sequence/:id

Tujuan utama dari *endpoint API* ini adalah untuk mengubah urutan *sequence* data keuntungan / *benefit* dari asuransi yang dipilih. Secara sederhana, alur jalannya program yang dibuat untuk mengubah urutan *sequence* data keuntungan asuransi di Tokopedia didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.45




Gambar 3.45. *Flowchart API /configuration/benefit/update-sequence/:id* di Modul Asuransi Penerbangan

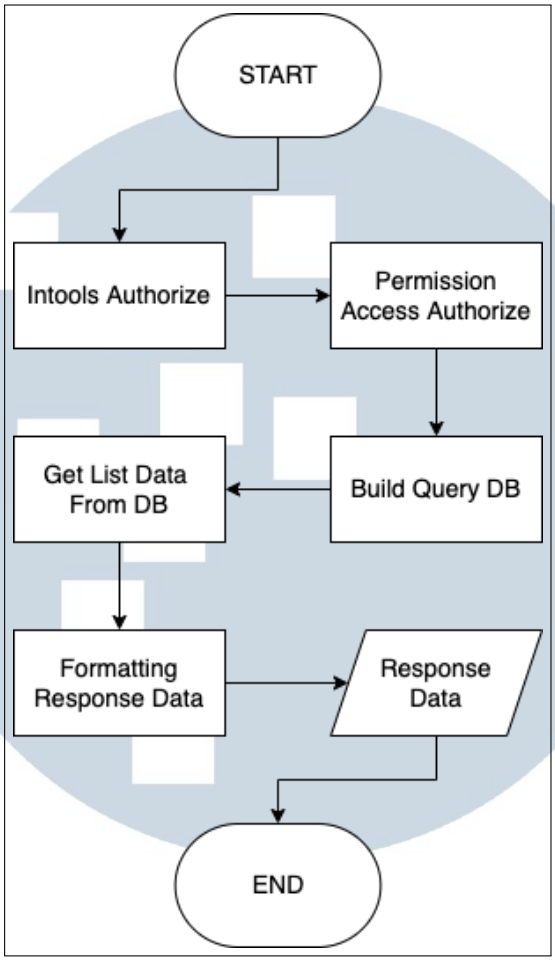
Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.46

```
type Response struct {
    Data struct {
        Success bool   `json:"Success"`
        Message string `json:"Message"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.46. *Struktur data response API /configuration/benefit/update-sequence/:id di Modul Asuransi Penerbangan*

ix. *Endpoint* /configuration/filter/partner

Tujuan utama dari *endpoint API* ini adalah untuk menampilkan dafatr data partner untuk kebutuhan filter / menyaring data konfigurasi asuransi di Tokopedia. Secara sederhana, alur jalannya program yang dibuat untuk menampilkan data filter asuransi ini didefinisikan dengan menggunakan *flowchart diagram* seperti pada Gambar 3.47




Gambar 3.47. *Flowchart API /configuration/filter/partner* di Modul Asuransi Penerbangan

Sedangkan struktur data yang digunakan sebagai *response* balikan dari *API* ini didefinisikan seperti pada Gambar 3.48

```
type Response struct {
    Data []struct {
        ID       int    `json:"id"`
        Name     string `json:"name"`
        ClientID string `json:"client_id"`
    } `json:"data"`
    Code    string `json:"code"`
    Latency string `json:"latency"`
}
```

Gambar 3.48. *Struktur data response API /configuration/filter/partner di Modul Asuransi Penerbangan*

### 3.3.2 Kendala yang Ditemukan

Kendala yang sering dialami selama proses kerja magang di Tokopedia adalah kendala teknikal diantaranya:

1. Adanya beberapa teknologi dan *tools* baru yang belum pernah ditemui dan digunakan sebelumnya seperti *monitoring tools*, *database replication system*, *queuing system*, dan lain lain, sehingga menghambat berlangsungnya proses kerja magang.

2. Kendala dalam mengelola data dalam jumlah yang cukup banyak, yang disebabkan karena kurangnya pengalaman dalam mengelola data yang banyak, sehingga dibutuhkan beberapa pertimbangan terkait *code* dan *query database* yang dibuat.

Namun kendala-kendala yang ditemukan selama proses kerja magang tentunya dapat diselesaikan dengan baik dengan caranya masing-masing.

### 3.3.3 Solusi Atas Kendala yang Ditemukan

Setiap kendala yang ditemukan pasti mempunyai solusinya masing-masing. Berikut solusi-solusi untuk kendala yang ditemukan selama proses kerja magang.

1. Kendala karena kurang familiarnya dengan teknologi dan *tools* baru yang digunakan di Tokopedia tentunya adalah dibutuhkannya waktu tambahan atau bahkan lembur di luar jam kerja untuk mempelajari hal-hal baru tersebut. Selain itu, mentor di Tokopedia juga selalu siap untuk membantu segala kendala yang ditemukan terkait masalah teknikal jika dibutuhkan.

2. Kendala karena kurangnya pengalaman mengelola data yang cukup banyak memiliki beberapa solusi seperti menulis *code* dengan efektif dan efisien atau tidak boros *resource*, menggunakan metode *asynchronous* untuk kondisi tertentu, dan mempelajari cara melakukan *query* ke *database* dengan tepat dan optimal. Selain itu juga ada diskusi dengan tim *Database administrators* dan *Database Engineers* terkait hal ini.