BAB 3

PELAKSANAAN KERJA MAGANG

3. 1 Kedudukan dan Koordinasi

Dalam pelaksanaan kerja magang di PT Indobest Artha Kreasi, jabatan yang dipegang adalah *Software Engineer Intern* pada tim API di divisi *Engineering* dan disupervisi langsung oleh Bapak Gandhi Pranata, selaku *Software Developer* pada tim API PT Indobest Artha Kreasi. Bapak Gandhi Pranata berperan dalam memberikan bimbingan dan informasi yang dibutuhkan selama pengembangan projek API *Report*. *Review* dan *feedback* juga diberikan secara berkala selama pelaksanaan kerja magang. Komunikasi selama kerja magang dilakukan melalui Telegram dan WhatsApp. Pertemuan progres dilaksanakan setiap hari selasa melalui Google Meet. Kolaborasi projek dan *code review* menggunakan aplikasi GitLab.

3. 2 Tugas yang Dilakukan

Berikut tugas – tugas yang dilakukan selama pelaksanaan kerja magang.

- Mengembangkan API *transaction report* untuk mengambil transaksi pulsa pra-bayar dalam bentuk JSON *pagination* dan file csv menggunakan Laravel PHP *framework* sesuai dengan dokumentasi yang telah diberikan.
- Membuat *testing* fitur dan unit (*controller*, *service* dan *repository test*) khususnya pada transaksi pra-bayar.

- Bekerja sama dengan anggota tim API lainnya dalam mencari dan menentukan metode terbaik untuk menyelesaikan masalah yang ditemukan selama pengembangan API *Report*.

3. 3 Uraian Pelaksanaan

3. 3. 1 Proses Pelaksanaan

Kerja magang dilaksanakan selama 2 bulan 2 minggu atau 10 minggu dengan *timeline* kerja sebagai berikut.

Tabel 3.1 Jadwal Kerja Magang

Kegiatan	Minggu									
	1	2	3	4	5	6	7	8	9	10
Setup *environment* dan Pengenalan alur kerja										
Briefing dan Inisialisasi projek API *Report*										
Implementasi fitur API *Report* (*Coding*)										
Membuat *feature* dan *unit testing*										
Refactor code pada beberapa aspek										
Bug fixing										

3. 3. 2 Uraian Alur Kerja Magang dan Framework

Pada minggu pertama dilakukan instalasi *software* dan *framework* yang diperlukan sepanjang kerja magang dan memahami struktur dan alur projek API yang sudah ada pada Mobilepulsa. *Software* yang digunakan adalah Visual Studio Code sebagai aplikasi *Code Editor* utama, Insomnia Core untuk melakukan hit API dan membantu dalam proses *debugging*, Git dan GitLab sebagai aplikasi pengontrol versi sistem (*Version Control*) dan *platform* kolaborasi dengan anggota lain tim API, dan Redis sebagai aplikasi penyimpanan data dalam bentuk *key* dan *value*. *Framework* yang digunakan dalam pembangunan sistem API adalah Laravel yang ditulis dalam bahasa pemrograman PHP.

Pada minggu kedua dilakukan *briefing* dan pembagian tugas pada projek API *Report*. Tugas yang diberikan adalah pengembangan API *Report* khususnya pada transaksi pulsa pra-bayar (*prepaid*). Dokumentasi API *Report* juga diberikan sebagai panduan utama dalam penyelesaian projek. Adapun *library* yang digunakan untuk mempermudah pengerjaan projek ini antara lain, Elasticquent yang digunakan untuk mengintegrasi Elasticsearch Eloquent dengan Laravel, Box/Spout yang merupakan *library* PHP yang membantu aplikasi membaca dan menulis pada file *spreadsheet* (CSV, XLSX dan ODS) secara cepat, dan Predis atau PHP Redis untuk mengintegrasi Redis dengan Laravel.

Elasticsearch merupakan *database* NoSQL yang berfokus pada mesin pencarian data (S. Amirlulah, 2019). Elasticsearch mendukung pencarian data untuk seluruh tipe data tak terkecuali tekstual, numerikal, *geospatial*, dan struktural data. Untuk melakukan operasi CRUD (*Create*, *Read*, *Update* dan *Delete*) pada suatu *database*, dapat dilakukan dengan mengirim *request* menuju API yang telah

dibuat oleh Elasticsearch disertai dengan parameter *index database* yang dituju. Terdapat perbedaan terminologi di antara Elasticsearch dan *relational database* (MySQL, Oracle) lainnya, seperti *index*, *type* dan *document*. *Index* pada Elasticsearch memiliki konsep yang sama dengan *Table* pada *relational database*. *Type* memiliki konsep yang sama dengan *Table*, sedangkan *Document* memiliki konsep yang sama dengan *Row* pada *relational database*. Elasticsearch memiliki kecepatan dan skalabilitas yang tinggi dibandingkan *relational database* pada umumnya.

Implementasi fitur API *Report* dilakukan mulai dari minggu ketiga hingga minggu terakhir kerja magang. Mengikuti struktur projek API lainnya yang sudah ada, struktur projek API *Report* terbagi menjadi beberapa bagian, *Controller*, *Service* dan *Repository*. *Controller* bertugas untuk melakukan validasi *parameter request* yang masuk mengikuti aturan yang sudah ditentukan, kemudian meneruskan *request* menuju *service*. *Service* bertugas untuk memproses *request* yang masuk, menjalankan tugas sesuai dengan *parameter request* dan mengembalikan respons kepada user dalam bentuk JSON atau XML. *Repository* berfokus pada segala hal yang berhubungan dengan *database*. Pada projek ini, *Repository* bertugas mengambil data transaksi dari *database* SQL atau indeks *elasticsearch* dan melakukan *field formatting* agar transaksi yang dikembalikan sesuai dengan dokumentasi.

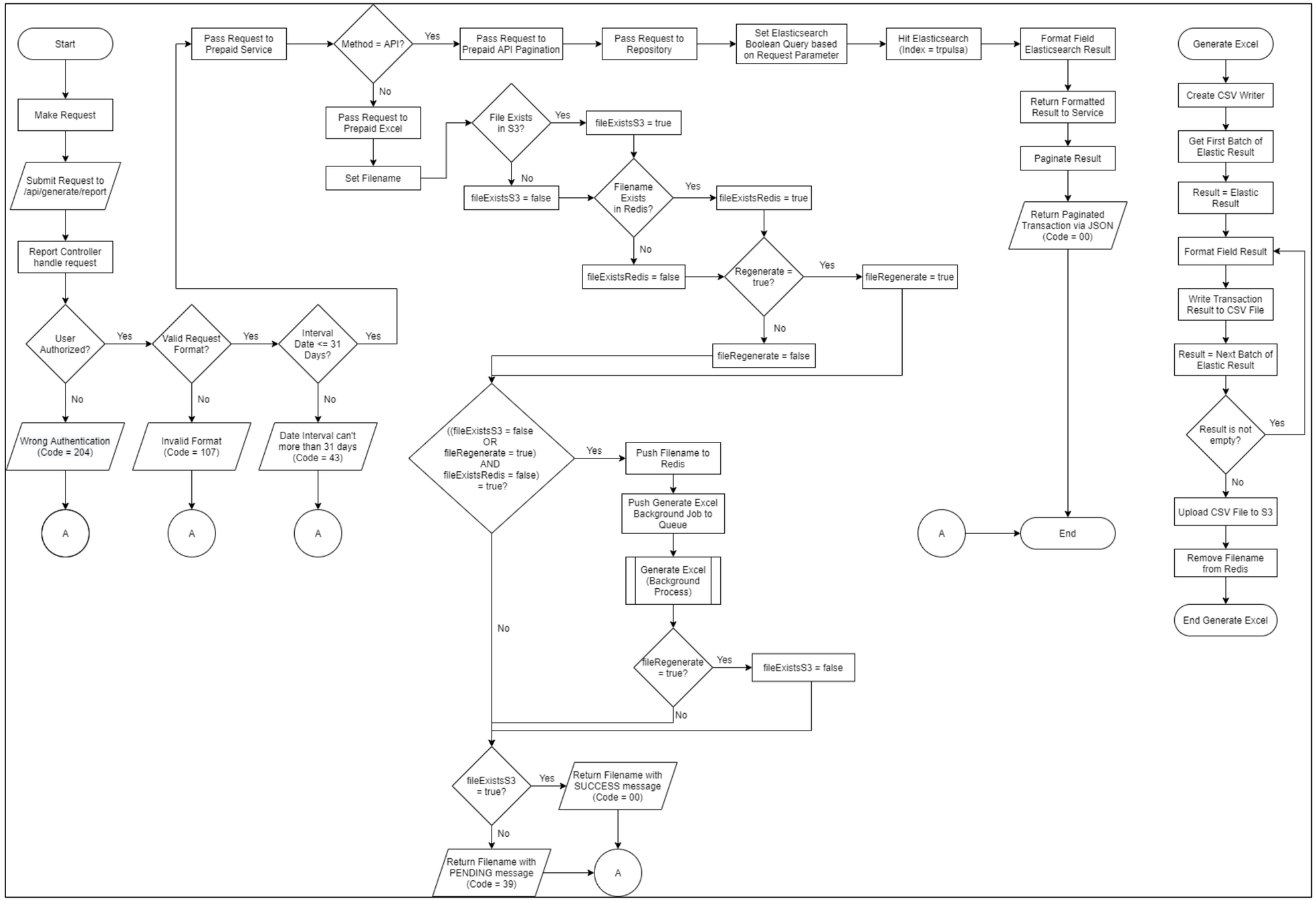
Pembuatan *feature* dan *unit testing* dimulai pada minggu keenam. *Testing* dibuat agar ketika aplikasi mencapai tahap *production* dan *release*, seluruh fitur dapat berjalan sebagaimana mestinya tanpa adanya *bug*. *Testing* dibuat menggunakan *framework* PHP Unit dengan bahasa pemrograman PHP. Fitur *test*

yang dibuat terdiri dari *Generate Report test* dan *Check Status test*. *Unit test* yang dibuat terdiri dari *Controller test*, *Service test* dan *Repository test*. Selama mengimplementasikan fitur dan *testing* pada projek API *Report*, dilakukan juga *code refactor* secara kontinu untuk menjaga kodingan rapi dan mudah dibaca oleh anggota tim lainnya.

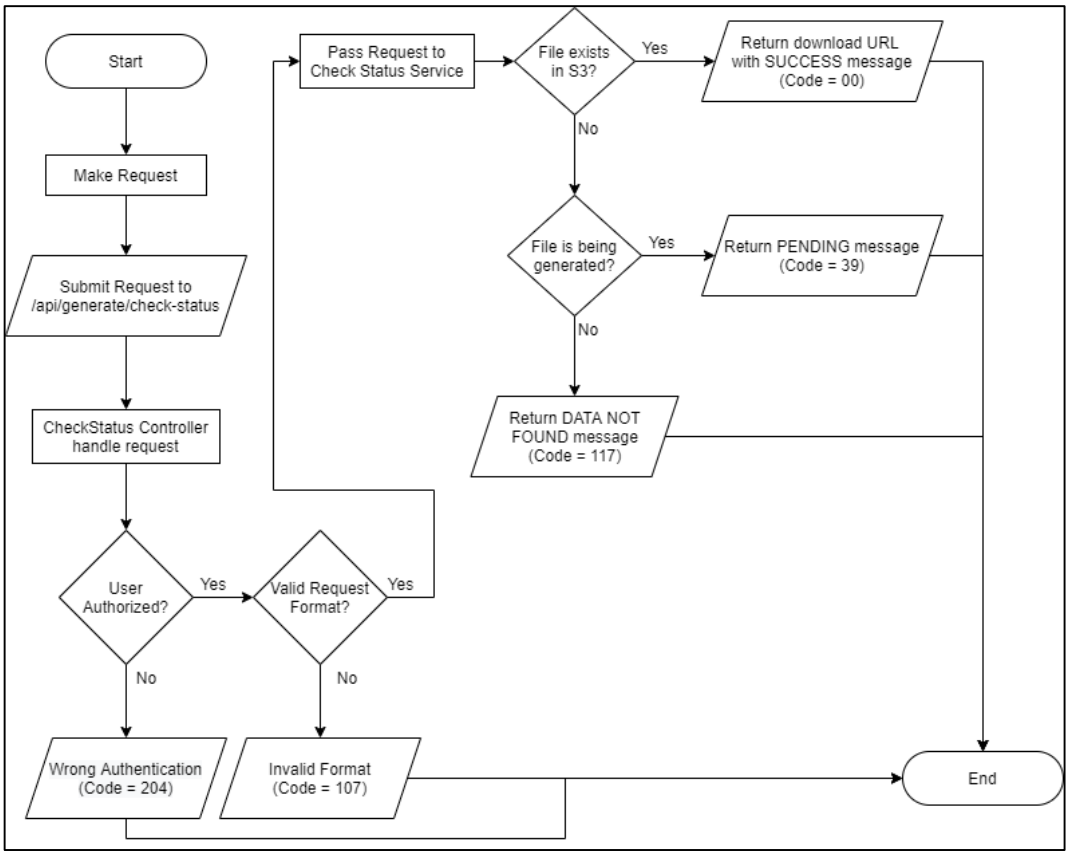
Untuk menyamakan struktur projek API yang sudah ada di Mobilepulsa, projek API *Report* dibangun dengan struktur kode berbasis objek. Agar struktur kodingan tetap bersih dan mudah dibaca, diterapkan prinsip SOLID yang dicetus oleh Robert C. Martin. Prinsip SOLID merupakan akronim dari 5 (lima) prinsip yaitu, *Single Responsibility Principle* (SPR), *Open/Closed Principle* (OCP), *Liskov Substitution Principle* (LSP), *Interface Segregation Principle* (ISP) dan *Dependency Inversion Principle* (DIP). SPR bermaksud bahwa setiap *class* dan *function* hanya bisa memiliki satu tujuan spesifik saja. OCP merupakan prinsip dimana setiap *class* dan *function* harus dapat terbuka untuk ekstensi (*Inheritance*) dan tertutup untuk modifikasi. LSP merupakan prinsip dimana objek yang merupakan turunan dari objek lain harus dapat menggantikan posisi objek induk. ISP merupakan prinsip dimana setiap servis tidak wajib untuk menggunakan fungsi yang tidak akan digunakan selama proses tertentu. Maka dari itu setiap kumpulan fungsi yang berkaitan satu sama lain akan dikumpulkan pada *interface* yang sama. DIP merupakan prinsip dimana suatu modul tingkat tinggi seharusnya tidak bergantung pada modul tingkat rendah karena akan mempersulit *programmer* jika harus melakukan perubahan pada modul tertentu yang erat dependensinya dengan modul lainnya.

3. 3. 3 Perancangan Sistem

A. Flowchart



Gambar 3.1 Flowchart *Generate Report*



Gambar 3.2 Flowchart *Check Status*

Gambar 3.1 merupakan *flowchart* alur proses fitur *generate report* pada projek API *Report*. *Flow generate report* dimulai dari user mengirim *request* yang valid kepada *end point generate report*. Kemudian *Controller* melakukan pengecekan *username*, format *request* dan interval waktu laporan transaksi. Setelah *request* user lolos pengecekan, *request* akan dikirimkan menuju servis. Pada servis, *request* akan dikirimkan lagi menuju sub-servis yang sesuai dengan *method* yang terdapat pada *request*. Jika *method* merupakan API, maka sistem akan menarik data transaksi dari elasticsearch sesuai dengan parameter *request* user dan melakukan *pagination* pada data tersebut. Kemudian API akan mengembalikan respons transaksi dengan format JSON dan pesan sukses kepada user. Jika *method* merupakan *file*, maka sistem pertama-tama akan mengecek ketersediaan *file* yang

ingin dibuat pada *queue* dan penyimpanan AWS S3. Jika *file* sudah tersedia dalam *queue* maupun penyimpanan AWS S3, maka sistem tidak akan menjalankan *job generate excel*. Jika *file* belum tersedia, maka sistem akan menjalankan *job generate excel* untuk membuat laporan transaksi dalam format CSV pada *background process* dan mengembalikan status *pending* kepada user.

Gambar 3.2 merupakan *flowchart* alur proses fitur *check status* pada projek API *Report*. *Flow check status* dimulai dari user mengirim *request* yang valid kepada *end point check status*. *Controller* juga akan melakukan validasi *username* dan pengecekan format *field request*. Jika *request* sudah sesuai, *request* akan dikirimkan menuju servis untuk dilakukan pengecekan nama file. Jika *file* yang diminta sudah selesai dibuat dan di-*upload* menuju penyimpanan AWS S3, maka API akan mengembalikan *download* url file transaksi yang diminta beserta pesan status sukses. Jika *file* yang diminta sedang dibuat dalam *queue*, maka API akan merespons dengan status *pending*. Jika *file* tidak berada di *queue* maupun S3, maka API akan merespons dengan status data tidak ditemukan.

B. Parameter Request dan Response

B. 1 Generate Report

Fitur *Generate Report* berfungsi untuk menghasilkan data transaksi pulsa pra-bayar dalam kurun waktu satu bulan dalam bentuk JSON, XML atau file CSV. URL untuk men-*download* transaksi pada file CSV dapat dilakukan pada fitur *Check Status*. *End point* untuk fitur ini adalah /api/report/generate.

B. 1. 1 Request Parameter

Tabel 3.2 merupakan tabel format *request* parameter yang valid pada fitur *Generate Report*.

Tabel 3.2 Tabel *Request Parameter* - *Generate Report*

Nama *Field*	Tipe Data	Bersifat Wajib	Deskripsi
username	string	Ya	Username yang sudah terdaftar
start_date	string	Ya	Tanggal mulai transaksi
end_date	string	Ya	Tanggal akhir transaksi
status	string	Ya	Status transaksi [‘all’, ‘success’, ‘pending’ or ‘failed’]
type	string	Ya	Tipe transaksi [‘prepaid’]
method	string	Ya	Format transaksi [‘api’ or ‘file’]
filter	array	Tidak	Filter transaksi berdasarkan *field* *Field* yang tersedia pada transaksi *prepaid* : - customer_id
sign	string	Ya	Hasil *hash* untuk autentikasi user

Tabel 3.2 Tabel *Request Parameter* - *Generate Report* (lanjutan)

Nama *Field*	Tipe Data	Bersifat Wajib	Deskripsi
regenerate	boolean	Tidak	Jika true, akan melakukan *re-generate report*. *Field* bersifat valid jika dan hanya jika method = 'file'

B. 1. 2 Contoh Request JSON

Gambar 3.3 dan 3.4 merupakan contoh bentuk JSON *request* valid yang diperlukan untuk menggunakan fitur *Generate Report.*

```
{
  "username"    : "<username>",
  "status"      : "all",
  "start_date"  : "2020-09-01",
  "end_date"    : "2020-09-30",
  "method"      : "api",
  "type"        : "prepaid",
  "sign"        : "<sign>"
}
```

Gambar 3.3 Valid JSON *Request Parameter*

```
{
  "username"    : "<username>",
  "status"      : "all",
  "start_date"  : "2020-09-01",
  "end_date"    : "2020-09-30",
  "method"      : "api",
  "type"        : "prepaid",
  "filter"      : [
    {
      "customer_id" : "<customer_id>"
    }
  ],
  "sign"        : "<sign>"
}
```

Gambar 3.4 Valid JSON *Request Parameter* dengan filter

B. 1. 3 Contoh Request XML

Gambar 3.5 dan 3.6 merupakan contoh bentuk XML *request* valid yang diperlukan untuk menggunakan fitur *Generate Report.*

```
<mp>
  <username>username</username>
  <status>all</status>
  <start_date>2020-09-01</start_date>
  <end_date>2020-09-30</end_date>
  <method>api</method>
  <type>prepaid</type>
  <sign>sign</sign>
</mp>
```

Gambar 3.5 Valid XML *Request Parameter*

```
<mp>
  <username>username</username>
  <status>all</status>
  <start_date>2020-09-01</start_date>
  <end_date>2020-09-30</end_date>
  <method>api</method>
  <type>prepaid</type>
  <filter>
    <customer_id>customer_id</customer_id>
  </filter>
  <sign>sign</sign>
</mp>
```

Gambar 3.6 Valid XML *Request Parameter* dengan filter

B. 1. 4 Response Parameter

Tabel 3.3 dan 3.4 merupakan tabel format *response* parameter yang valid pada fitur *Generate Report*.

1) Response JSON Pagination

Tabel 3.3 Tabel *Response Parameter* JSON - *Generate Report*

Nama *Field*	Tipe Data	Deskripsi
current_page	integer	Halaman sekarang
total_page	integer	Total halaman transaksi
total_data	integer	Jumlah transaksi yang dikembalikan
report	array of object	Kumpulan transaksi dalam bentuk *array*
report[*][datetime]	string	Waktu transaksi
report[*][tr_id]	integer	ID transaksi
report[*][customer_id]	string	No Hp tujuan transaksi
report[*][product_code]	string	Kode produk transaksi
report[*][nominal]	string	Denom transaksi
report[*][price]	integer	Biaya transaksi
report[*][status]	string	Status transaksi
report[*][sn]	string	Nomor serial transaksi
report[*][ref_id]	string	ID referens transaksi
next_page_url	string	URL menuju halaman transaksi berikutnya
prev_page_url	string	URL menuju halaman transaksi sebelumnya
rc	string	Kode respons
message	string	Pesan respons

2) Response File

Tabel 3.4 Tabel *Response Parameter* File - *Generate Report*

Nama *Field*	Tipe Data	Deskripsi
filename	string	Nama file transaksi
rc	string	Kode respons
message	string	Pesan respons

B. 1. 5 Contoh Response JSON

Gambar 3.7 dan 3.8 merupakan contoh bentuk JSON *response* valid pada fitur *Generate Report.*


```
{
  "data": {
    "current_page": 1,
    "total_page": 1,
    "total_data": 2,
    "report": [
      {
        "datetime": [redacted],
        "tr_id": [redacted],
        "customer_id": [redacted],
        "product_code": [redacted],
        "nominal": [redacted],
        "price": [redacted]
        "status": [redacted],
        "sn": [redacted],
        "ref_id": [redacted]
      },
      {
        "datetime": [redacted],
        "tr_id": [redacted],
        "customer_id": [redacted],
        "product_code": [redacted],
        "nominal": [redacted],
        "price": [redacted],
        "status": [redacted],
        "sn": [redacted],
        "ref_id": [redacted]
      }
    ],
    "next_page_url": null,
    "prev_page_url": null,
    "rc": "00",
    "message": "SUCCESS"
  }
}
```


Gambar 3.7 JSON *Response Parameter* – API

```
{
  "data": {
    "filename": [redacted],
    "rc": "00",
    "message": "SUCCESS"
  }
}
```


Gambar 3.8 JSON *Response Parameter* - File

B. 1. 6 Contoh Response XML

Gambar 3.9 dan 3.10 merupakan contoh bentuk XML *response* valid pada fitur *Generate Report.*


```
<?xml version="1.0"?>
<mp>
  <current_page>1</current_page>
  <total_page>1</total_page>
  <total_data>2</total_data>
  <report>
    <datetime>[redacted]</datetime>
    <tr_id>[redacted]</tr_id>
    <customer_id>[redacted]</customer_id>
    <product_code>[redacted]</product_code>
    <nominal>[redacted]</nominal>
    <price>[redacted]</price>
    <status>[redacted]</status>
    <sn></sn>
    <ref_id>[redacted]</ref_id>
  </report>
  <report>
    <datetime>[redacted]</datetime>
    <tr_id>[redacted]</tr_id>
    <customer_id>[redacted]</customer_id>
    <product_code>[redacted]</product_code>
    <nominal>[redacted]</nominal>
    <price>[redacted]</price>
    <status>[redacted]</status>
    <sn></sn>
    <ref_id>[redacted]</ref_id>
  </report>
  <next_page_url></next_page_url>
  <prev_page_url></prev_page_url>
  <rc>00</rc>
  <message>SUCCESS</message>
</mp>
```


Gambar 3.9 XML *Response Parameter* - API

```
<?xml version="1.0"?>
<mp>
  <filename>[redacted]</filename>
  <rc>00</rc>
  <message>SUCCESS</message>
</mp>
```

Gambar 3. 10 XML *Response Parameter* - File

B. 2 Check Status

Fitur *Check Status* berfungsi untuk melakukan pengecekan status file transaksi yang sudah di-*generate* pada fitur *Generate Report* sebelumnya. URL untuk men-*download* transaksi akan muncul dalam respons jika dan hanya jika status yang dikembalikan adalah *success*. *End point* untuk fitur ini adalah /api/report/check-status.

B. 2. 1 Request Parameter

Tabel 3.5 merupakan tabel format *request* parameter yang valid pada fitur *Check Status*.

Tabel 3.5 Tabel *Request Parameter – Check Status*

Nama *Field*	Tipe Data	Bersifat Wajib	Deskripsi
username	string	Ya	Username yang sudah terdaftar
filename	string	Ya	Nama file transaksi
sign	string	Ya	Hasil *hash* untuk autentikasi user

B. 2. 2 Contoh Request JSON

Gambar 3.11 merupakan contoh bentuk JSON *request* valid yang diperlukan untuk menggunakan fitur *Check Status*.

```
{
  "filename" : "<filename>",
  "username" : "<username>",
  "sign"     : "<sign>"
}
```

Gambar 3.11 Valid JSON *Request Parameter* - *Check Status*

B. 2. 3 Contoh Request XML

Gambar 3.12 merupakan contoh bentuk XML *request* valid yang diperlukan untuk menggunakan fitur *Check Status.*

```
<mp>
  <filename>filename</filename>
  <username>username</username>
  <sign>sign</sign>
</mp>
```

Gambar 3.12 Valid XML *Request Parameter* - *Check Status*

B. 2. 4 Response Parameter

Tabel 3.6 merupakan tabel format *response* parameter yang valid pada fitur *Check Status.*

Tabel 3.6 Tabel *Response Parameter – Check Status*

Nama *Field*	Tipe Data	Deskripsi
filename	string	Nama file transaksi
downloadUrl	string	Url untuk men-*download* file transaksi (hanya tersedia jika message = 'SUCCESS')
rc	string	Kode respons
message	string	Pesan respons

B. 2. 5 Contoh Response JSON

Gambar 3.13 merupakan contoh bentuk JSON *response* valid pada fitur *Check Status.*

```
{
  "data": {
    "filename": [redacted],
    "downloadUrl": [redacted],
    "rc": "00",
    "message": "SUCCESS"
  }
}
```

Gambar 3.13 Valid JSON *Response Parameter* - *Check Status*

B. 2. 6 Contoh Response XML

Gambar 3.14 merupakan contoh bentuk XML *response* valid pada fitur *Check Status.*

```
<?xml version="1.0"?>
<mp>
  <filename>[redacted]</filename>
  <downloadUrl>[redacted]</downloadUrl>
  <rc>00</rc>
  <message>SUCCESS</message>
</mp>
```

Gambar 3.14 Valid XML *Response Parameter* - *Check Status*

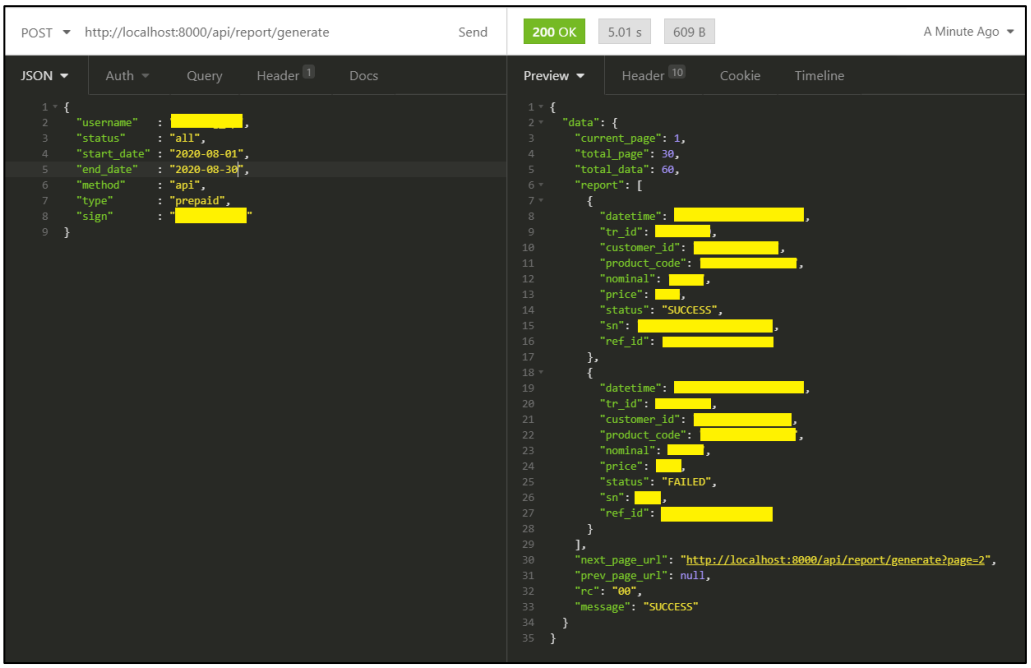
3. 3. 4 Implementasi

Pengujian API dilakukan dengan aplikasi Insomnia Core. Pengujian dilakukan secara local dengan domain *localhost*. *Database* yang digunakan MySQL untuk mengautentikasi *user* dan Elasticsearch sebagai tempat penyimpanan transaksi pulsa pra-bayar. Autentikasi API dilakukan dengan pengecekan parameter *sign* pada *request* yang berisi fungsi *hash* MD5 dari *username* + API *key* + format servis yang digunakan ('prepaid'). Pengujian API dilakukan pada fitur *generate report* dan *check status* dengan mempertimbangkan seluruh skenario yang dapat terjadi. Beberapa *value* pada *body request* dan *response* gambar hasil pengujian akan disensor demi menjaga rahasia perusahaan.

A. Pengujian fitur *Generate Report*

1. Hasil pengujian API JSON *pagination*

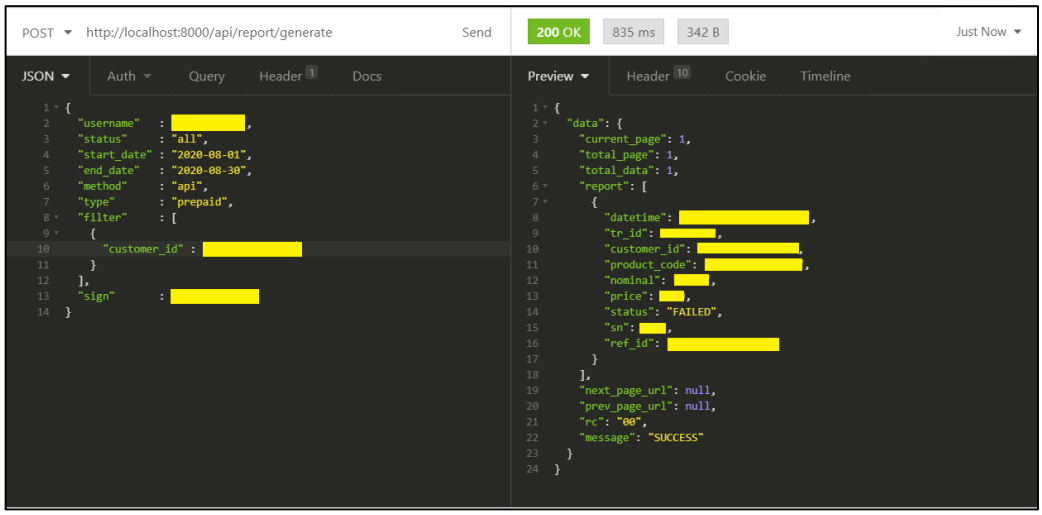
Pada Gambar 3.15 terdapat gambar *request* beserta *response* yang diberikan dalam bentuk JSON *pagination*. Parameter *username* pada *request* menandakan nomor hp asal atau nomor yang melakukan transaksi. Maka dari itu, API hanya akan mengembalikan transaksi yang dilakukan nomor hp asal tersebut. Transaksi juga akan disaring sesuai dengan parameter *request* lainnya seperti *filter* dan *status* dalam kurun waktu di antara parameter tanggal mulai dan tanggal akhir. Url pada parameter 'next_page_url' memiliki parameter *page*=2 yang menandakan halaman kedua laporan transaksi yang dikembalikan API untuk *request* terkait. Jika parameter *page* tidak ada, maka API akan mengembalikan halaman pertama transaksi.



Gambar 3.15 Hasil Pengujian API JSON *Pagination*

2. Hasil pengujian API JSON *pagination* dengan filter

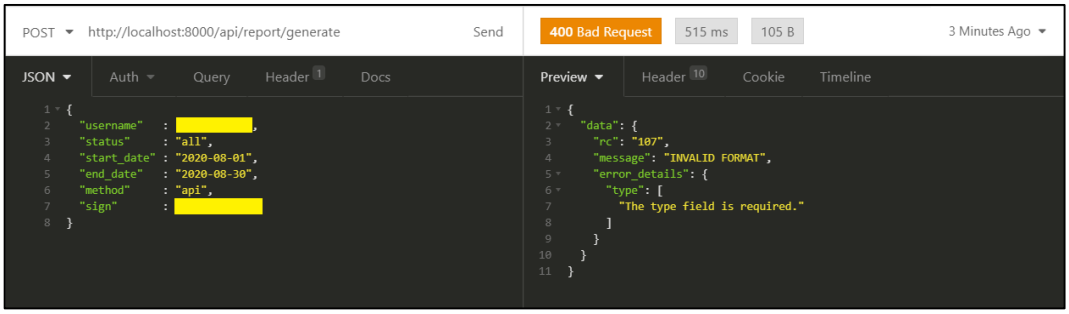
Pada Gambar 3.16 terdapat gambar *request* beserta *response* yang diberikan dalam bentuk JSON *pagination* dengan parameter *filter*. Parameter ‘customer_id’ merupakan nomor hp tujuan transaksi. API hanya akan mengembalikan transaksi dengan nomor hp tujuan sesuai dengan *value* ‘customer_id’ pada *filter*.



Gambar 3.16 Hasil Pengujian API JSON *Pagination* dengan *filter*

3. Hasil pengujian API dengan *request* yang tidak valid

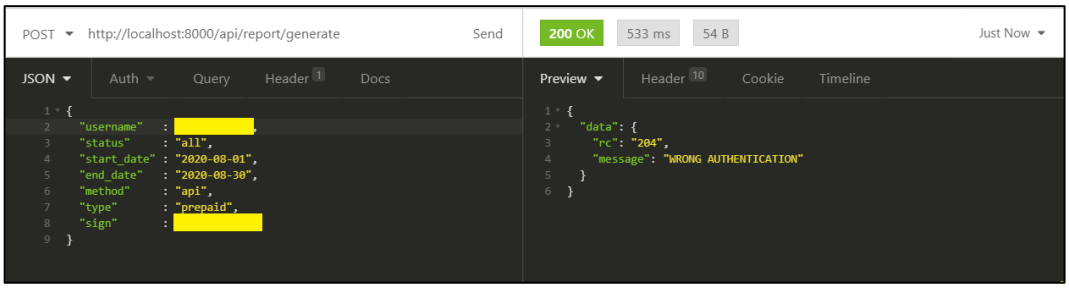
Pada Gambar 3.17 terdapat hasil pengujian API dengan *body request* tanpa *field* ‘type’. API akan mengembalikan respons ‘400 Bad Request’ dengan message ‘INVALID FORMAT’ beserta detail *error*.



Gambar 3.17 Hasil Pengujian API (*Invalid Format*)

4. Hasil pengujian API dengan *sign* yang salah

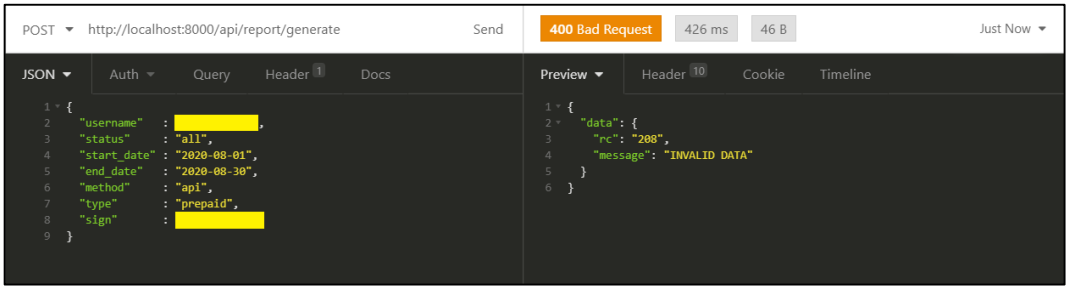
Pada Gambar 3.18 terdapat hasil pengujian API dengan *sign* yang salah. API akan mengembalikan respons dengan pesan ‘WRONG AUTHENTICATION’ yang menandakan kesalahan *value sign* untuk *username* terkait.



Gambar 3.18 Hasil Pengujian API (*Wrong Authentication*)

5. Hasil pengujian API dengan *username* yang tidak terdaftar

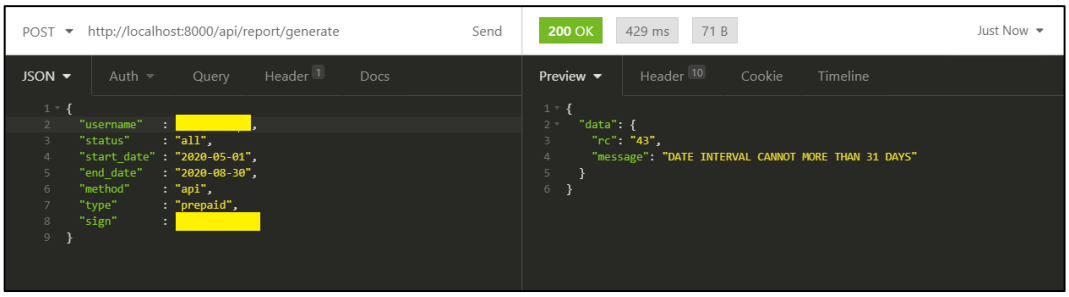
Pada Gambar 3.19 terdapat hasil pengujian API dengan *username* yang tidak terdaftar. API akan mengembalikan respons ‘400 Bad Request’ dengan pesan ‘INVALID DATA’.



Gambar 3.19 Hasil Pengujian API (*Invalid Data*)

6. Hasil pengujian API dengan interval waktu lebih dari 31 hari

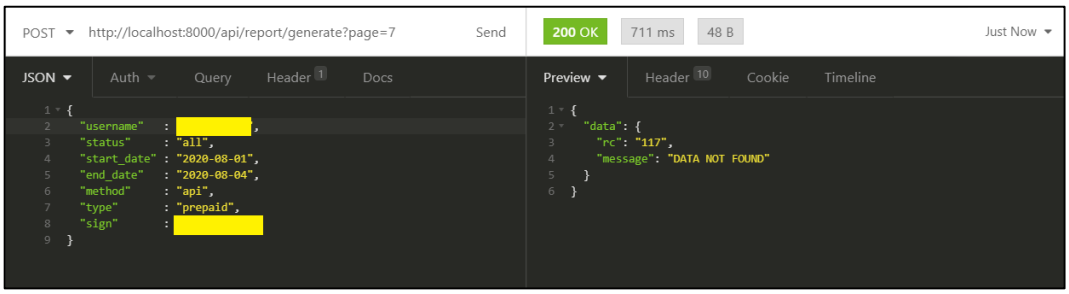
Pada Gambar 3.20 terdapat hasil pengujian API dengan selisih ‘start_date’ dan ‘end_date’ lebih dari 31 hari. API akan mengembalikan respons dengan pesan ‘DATE INTERVAL CANNOT MORE THAN 31 DAYS’.



Gambar 3.20 Hasil Pengujian API (*Invalid Date Interval*)

7. Hasil pengujian API dengan parameter *page* melebihi total *page*

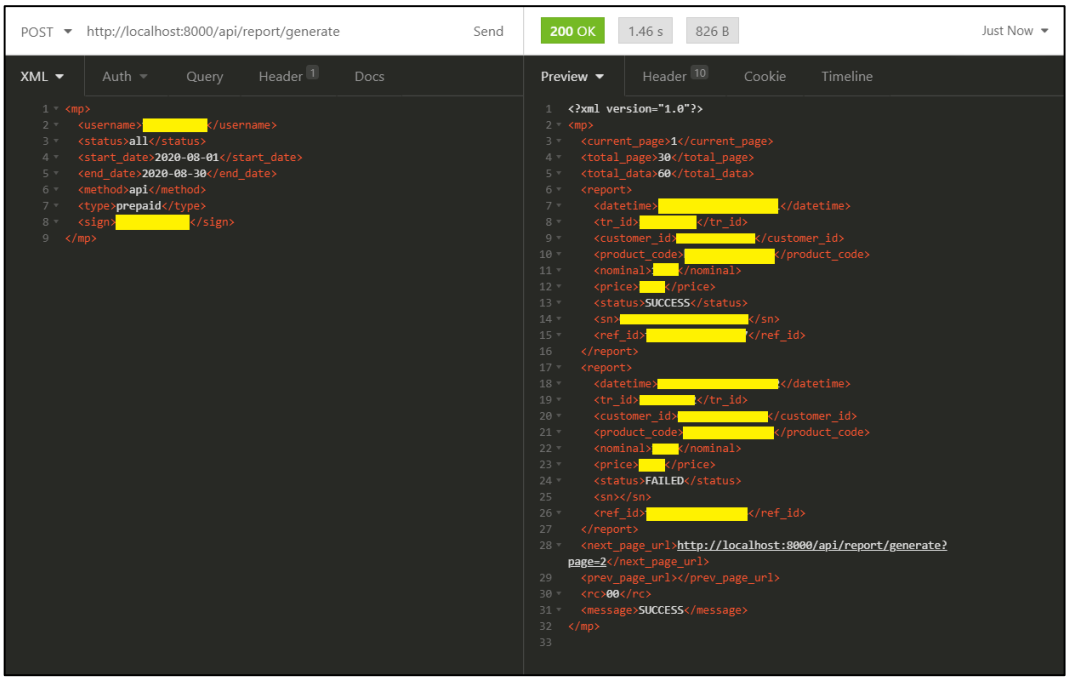
Pada Gambar 3.21 terdapat hasil pengujian API dengan parameter *page* melebihi total *page* yang ada. API akan mengembalikan respons dengan pesan 'DATA NOT FOUND. Parameter *page* adalah 7 (tujuh) sedangkan total *page* hanya ada 5 (lima) buah.



Gambar 3.21 Hasil Pengujian API (*Data Not Found*)

8. Hasil pengujian API dengan respons XML

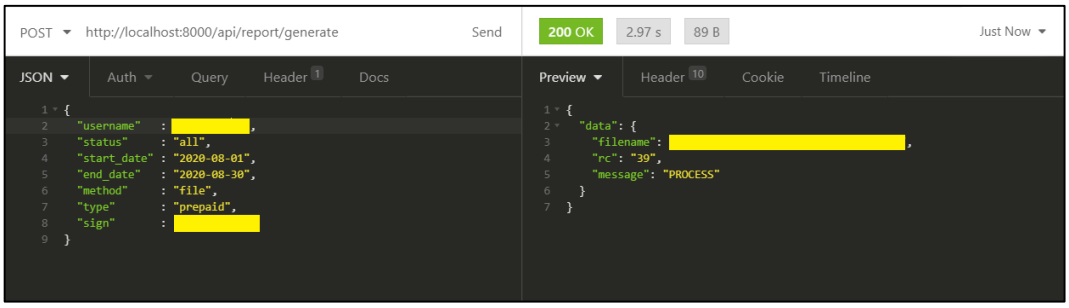
Pada Gambar 3.22 terdapat hasil pengujian API dengan respons XML. API mengembalikan transaksi disertai pesan 'SUCCESS'.



Gambar 3.22 Hasil Pengujian API dengan respons XML

9. Hasil pengujian API dengan metode file

Pada Gambar 3.23 terdapat hasil pengujian API dengan metode file. API mengembalikan respons nama file transaksi beserta pesan ‘PROCESS’. API akan membuat laporan transaksi dalam bentuk file CSV. Proses *generate* file CSV merupakan proses *background*.

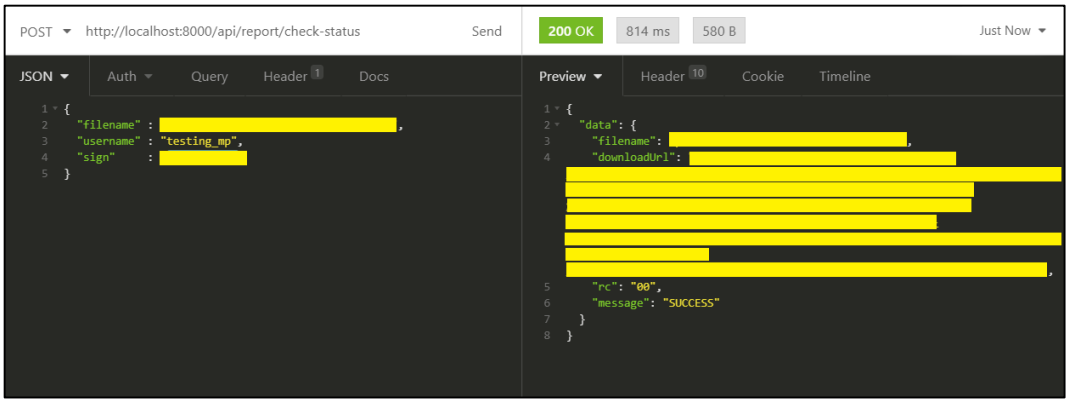


Gambar 3.23 Hasil pengujian API dengan metode file

B. Pengujian fitur *Check Status*

1. Hasil pengujian API *Check Status* dengan respons sukses

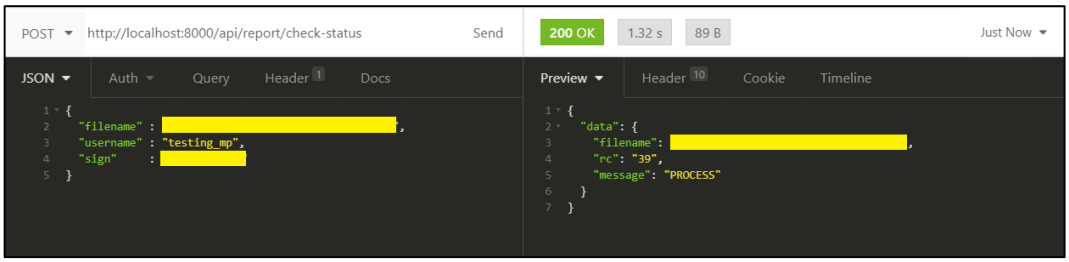
Pada Gambar 3.24 terdapat hasil pengujian API *Check Status* dengan status sukses. Pada kasus ini, file transaksi dengan nama *filename* sudah berhasil dibuat oleh fitur *generate report* sebelumnya dan berhasil di-*upload* menuju penyimpanan AWS S3. Maka API akan merespons dengan *field* 'downloadUrl' yang berisi url untuk men-*download* file transaksi disertai dengan pesan 'SUCCESS'.



Gambar 3.24 Hasil Pengujian API *Check Status* dengan respons sukses

2. Hasil pengujian API *Check Status* dengan respons *pending*

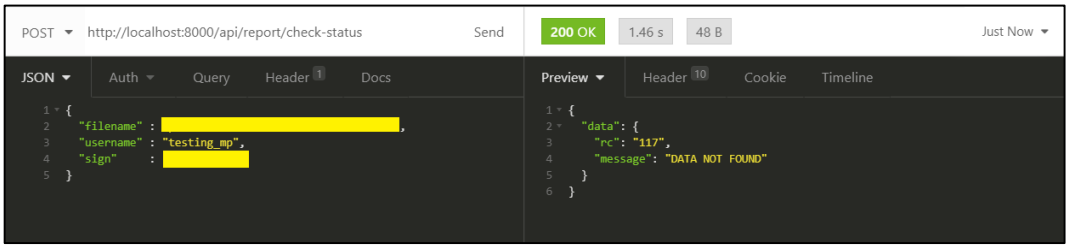
Pada Gambar 3.25 terdapat hasil pengujian API *Check Status* dengan status *pending*. Pada kasus ini, file transaksi dengan nama *filename* sedang berada pada *queue* pembuatan file oleh fitur *generate excel*. Maka API akan merespons dengan pesan 'PROCESS' tanpa *field* 'downloadUrl' selama file yang bersangkutan belum berada pada penyimpanan AWS S3.



Gambar 3.25 Hasil Pengujian API *Check Status* dengan respons *pending*

3. Hasil pengujian API *Check Status* dengan respons *data not found*

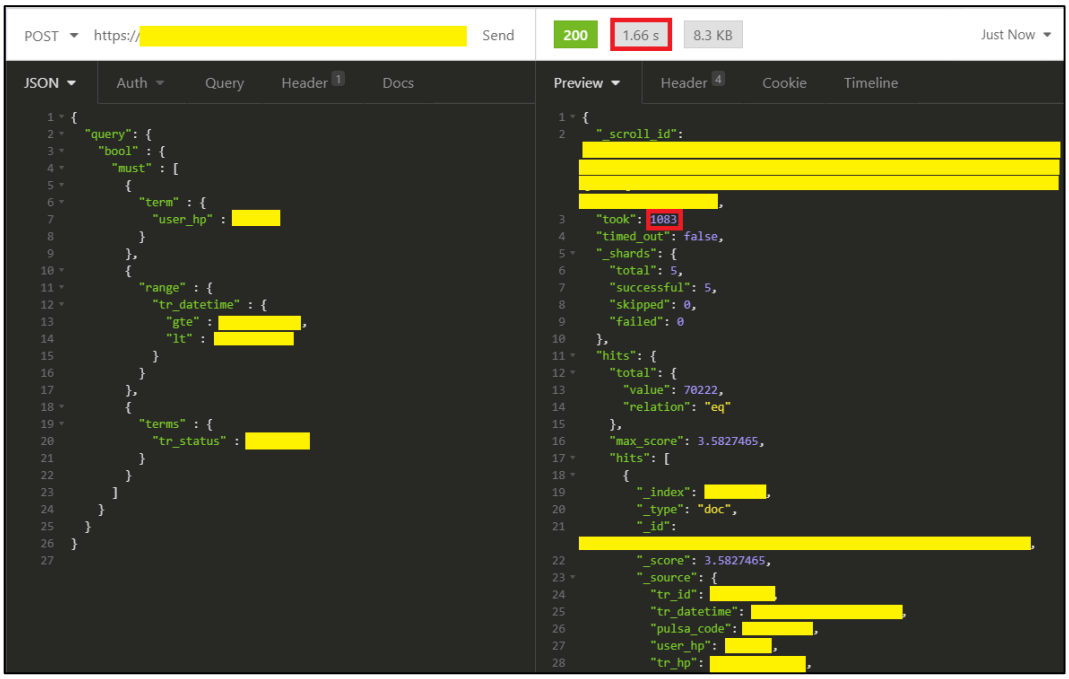
Pada Gambar 3.26 terdapat hasil pengujian API *Check Status* dengan status *data not found*. Pada kasus ini, file transaksi dengan nama *filename* belum pernah dibuat oleh fitur *generate report* sebelumnya dan tidak berada pada penyimpanan AWS S3. Maka API akan merespons dengan pesan 'DATA NOT FOUND' tanpa *field* 'downloadUrl'.



Gambar 3.26 Hasil Pengujian API *Check Status* dengan respons *data not found*

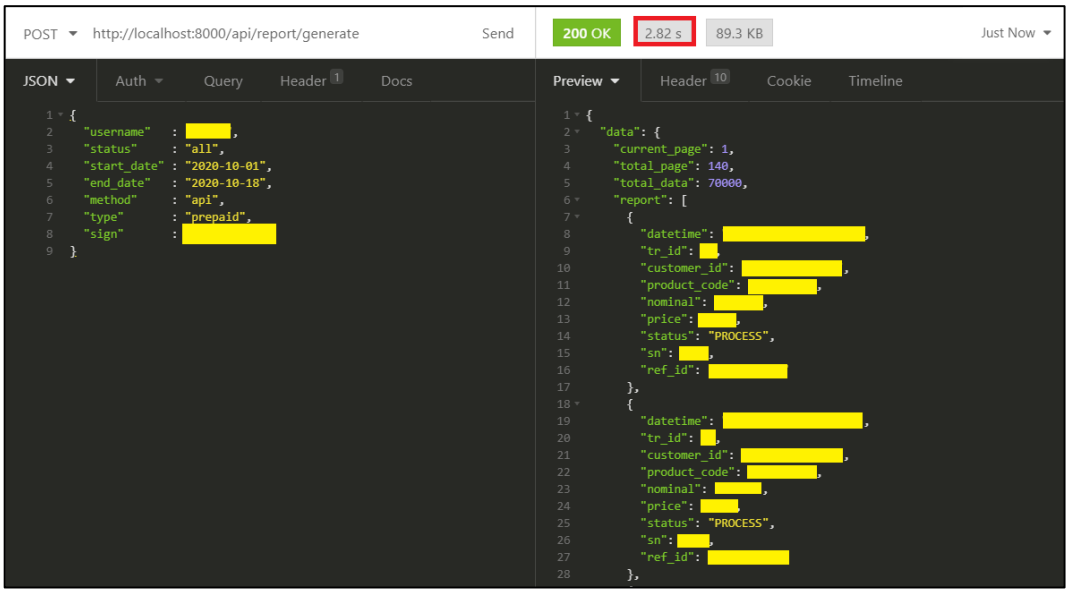
C. Perbandingan Waktu Pencarian Elasticsearch dan MySQL

Perbandingan waktu pencarian Elasticsearch dan MySQL dilakukan dengan menggunakan kurang lebih 70000 data transaksi. Gambar 3.27 merupakan hasil pengujian waktu pencarian menggunakan API Elasticsearch. *Query time* menggunakan Elasticsearch membutuhkan waktu 1,083 detik, sedangkan respons penuh dikembalikan dalam waktu 1,66 detik.



Gambar 3.27 Hasil Pengujian API Elasticsearch

Pada Gambar 3.28 terdapat hasil pengujian waktu pencarian menggunakan *query database* MySQL. Waktu yang diperlukan MySQL untuk mengambil 70000 data transaksi adalah 2.82 detik. Berdasarkan Gambar 3.27 dan 3.28, dapat disimpulkan bahwa *query time* Elasticsearch jauh lebih sedikit dibandingkan *database* MySQL terutama dalam memproses data dengan jumlah banyak. Penggunaan Elasticsearch tentunya memiliki dampak positif pada perusahaan terutama dalam aspek pemrosesan *queue*. Sistem yang dapat memproses *queue* dengan cepat tentunya dapat menampung lebih banyak *request* yang masuk dari klien.



Gambar 3.28 Hasil Pengujian Database MySQL

3. 3. 5 Kendala yang Ditemukan

Selama mengembangkan API *Report*, terdapat kendala – kendala yang ditemui dari segi teknis maupun non-teknis. Secara umum kendala – kendala berikut antara lain,

a. Seluruh komunikasi dan *progress meeting* dilakukan secara *online* atau jarak jauh sehingga sedikit terhambat pada saat diskusi suatu masalah. Selain itu, respons yang didapatkan selama berkomunikasi (*chat*) dengan tim tidak selalu instan.

b. Penerapan prinsip SOLID dalam penulisan kode pada projek API *Report* sedikit terhambat karena tidak terbiasa menulis kode program dengan *object oriented style*.

c. Pembuatan *testing* fitur atau unit yang melibatkan penarikan data dari *database* MySQL sedikit terhambat dikarenakan perbedaan *environment* projek di lokal dengan *environment* pada *pipeline* Jenkins.

3. 3. 6 Solusi Atas Kendala yang Ditemukan

Solusi untuk mengatasi kendala – kendala yang ditemukan selama kerja magang adalah sebagai berikut.

a. Mengatur jadwal *online meet* dengan orang yang bersangkutan jika masalah yang ditemukan cukup besar atau rumit untuk diselesaikan seorang diri.
b. Menerapkan prinsip SOLID sendiri adalah dengan mencoba menulis kode dengan beberapa cara dan membandingkan tingkat keterbacaan atau *readability*-nya.
c. Melakukan komunikasi dengan pihak yang mempersiapkan *pipeline* Jenkins pada projek yang bersangkutan untuk mendapatkan detail *environment* di Jenkins. Pada kasus yang dialami, *database* di lokal dan Jenkins tidak sama sehingga digunakan fitur *Factory* dari Laravel untuk memasukkan data sementara yang diperlukan selama *testing* dilakukan.