# BAB III

# PELAKSANAAN KERJA MAGANG

## 3.1 Kedudukan dan Koordinasi

Pelaksanaan kerja magang sebagai *Back End Developer* dilakukan dibawah bimbingan Christian Wijasa selaku *Assistant Engineering Manager* Connexi. Selama pengembangan *error mapping* dan pembuatan integrasi pelayanan baru untuk proses *reconnect*, penulis berkoordinasi dengan tim Connexi dengan menggunakan aplikasi *slack* serta melakukan pertemuan *online* setiap hari sebelum memulai kerja untuk menjabarkan tugas yang akan dikerjakan setiap harinya. Selain dengan pembimbing lapangan, penulis juga melakukan koordinasi dengan Muhammad Rizqi dan Galuh Octavia selaku *Product Owner,* Natasha Cindy selaku *Quality Assurance*, serta Bapak Wiguno selaku *Head of engineering*.

## 3.2 Tugas yang dilakukan

Selama pelaksanaan kerja magang, tugas dan tanggung jawab yang diberikan adalah sebagai berikut :

1. Melakukan *Scrapping* dokumentasi *error* pada web Blibli dan Tokopedia dengan metode cron menggunakan nodeJS.
2. Melakukan pembenahan struktur *error mapping* pada *package* Blibli dan Tokopedia di web Connexi.
3. Membuat *endpoint* reconnect saleschannel untuk *orchestrator* Connexi.
4. Membuat *test case* dan dokumentasi terhadap setiap komponen yang dibuat.

5. Mengembangkan website Connexi bersama tim serta membantu mengatasi bug yang ada.

## 3.3 Uraian Kerja Magang

### 3.3.1 Proses Pelaksanaan

Kerja magang yang dilaksanakan setiap minggunya diuraikan dalam *timeline* kerja sebagai berikut.

Tabel 3.1 Jadwal Pelaksanaan Kerja Magang

| Kegiatan yang dilakukan | Minggu ke- | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Melakukan instalasi *environment* Connexi dan mempelajari Bahasa pemrograman golang | | | | | | | | |
| Mempelajari struktur *database* dan melakukan *migration database* | | | | | | | | |
| Melakukan *scrapping error code* pada website Tokopedia dan Blibli menggunakan nodejs | | | | | | | | |
| Melakukan *revamp* untuk *error mapping* pada website Connexi | | | | | | | | |
| Membuat *test case* untuk *error mapping* | | | | | | | | |
| Membuat *endpoint* reconnect saleschannel untuk *orchestrator* Connexi | | | | | | | | |
| Membuat *test case* untuk *endpoint* reconnect saleschannel | | | | | | | | |
| *Bug fix testing* | | | | | | | | |

Pada minggu pertama, dilakukan instalasi *environment* Connexi mulai dari *github*, *Docker, Arcanist,* dan beberapa hal lain seperti akses ke *phabricator* SIRCLO dan *staging* Connexi. Selain itu, hal yang dilakukan adalah mempelajari Bahasa pemrograman Golang serta struktur *code* pada Connexi.

Pada minggu kedua, hal yang dilakukan adalah mempelajari struktur database Connexi. Database yang digunakan adalah PostgreSQL. Selain itu, dilakukan juga *migration database* di tabel account, store, warehouse untuk menambahkan *field* baru pada database yang digunakan untuk pengembangan website Connexi. Pada minggu kedua ini juga diadakan pertemuan via *online* dengan pembimbing untuk membahas projek yang akan dikerjakan selanjutnya.

Pada minggu ketiga, hal yang dilakukan adalah mempelajari *scrapping* menggunakan nodejs. *Scrapping* yang dilakukan dimaksudkan untuk mengambil data *error code* beserta deskripsinya pada website Tokopedia dan Blibli. *Scrapping* dilakukan dengan menggunakan *library* Puppeteer dengan metode *cron job.* Tujuan dari *scrapping* ini adalah untuk merubah sistem *error mapping* pada Connexi saat terjadi pemanggilan API ke *marketplace*. Pembuatan metode *scrapping* ini berlangsung sampai minggu keempat

Pada minggu keempat, selain pembuatan metode *scrapping*, dilakukan pembenahan untuk sistem pemanggilan *error* di Connexi. Pada Connexi, dilakukan *hardcode* untuk *error code* dan deskripsinya. Alasan dibuatnya *hardcode* pada Connexi antara lain agar meringankan kerja sistem serta alasan keamanan sistem sendiri.

Pada minggu kelima, hal yang dilakukan adalah memastikan *final* error mapping pada Connexi dari *review* yang diusulkan tim. Selain itu, dibuat juga test case pada Connexi untuk mengecek *code error* dan deskripsi yang didapat saat terjadi kegagalan pemanggilan API ke *marketplace*.

Pada minggu keenam dan ketujuh, hal yang dilakukan adalah membuat *endpoint* untuk *orchestrator* baru Connexi yaitu JSON-Rpc. *Endpoint* yang dikerjakan adalah *reconnect* saleschannel, untuk penghubungan ulang saleschannel. Selain itu, dibuat juga test case untuk memastikan proses *reconnect* berhasil.

Pada minggu kedelapan, dilakukan *review* setiap progress yang dikerjakan dengan pembimbing lapangan. Selain itu, dilakukan perbaikan *bug* atau *error* yang ditemukan pada sistem. Perbaikan yang dilakukan seperti terjadi kegagalan pada saat dilakukan test di *staging* atau pembenahan *log* yang dibuat.

### 3.3.2 Tools dan Requirement yang digunakan

Pada pelaksanaan kerja magang, terdapat beberapa tools yang digunakan untuk menunjang kerja magang. Untuk proses pembagian tugas dan koordinasi dalam tim digunakan Phabricator dan Arcanist. Phabricator digunakan untuk pembentukan *sprint planning*, pembagian tugas masing-masing anggota tim, serta merupakan tempat untuk melakukan *review code*. Sedangkan Arcanist merupakan *tool* yang digunakan untuk membuat *diffusion* baru dari suatu *task*, *commit* pada repositori git Connexi. Sebelum dilakukan *commit* terhadap suatu *diff*, terlebih dahulu harus dilakukan *review* oleh minimal tiga anggota tim hingga *diff* tersebut

diterima. Selain itu digunakan Nginx sebagai *web server* dengan konfigurasi sistem operasi Linux serta Docker yang digunakan untuk menjalankan program di *web server*.

Setiap *task* yang dikerjakan di repositori Connexi menggunakan Bahasa pemrograman Golang dengan *database* yang digunakan adalah PostgreSQL. Sementara untuk proses *scrapping* yang dilakukan diluar repositori Connexi menggunakan nodejs dengan library yang dipakai yaitu puppeteer. Alasan penggunaan puppeteer adalah karena *library* ini menggunakan Chrome sebagai *browser* dan mudah dalam mengekstrak data dari suatu web. Selain itu, digunakan juga tool reetro yang digunakan untuk evaluasi setiap *sprint* serta *google meet* untuk melakukan rapat atau pertemuan harian.

Terdapat beberapa *requirement* yang digunakan dalam pengerjaan *error mapping* dan *reconnect saleschannel. Requirement* yang dimaksud antara lain :

- Membuat Repositori baru untuk Scrapping Code dan Deskripsi Error pada web Tokopedia dan Blibli. Pada Repositori ini juga, dilakukan konfigurasi *cron job* untuk proses scrapping.
- Pada Repositori Connexi, dibuat sebuah *interface* baru dan dilakukan pembenahan struktur error mapping pada *package* Tokopedia dan Blibli. Selain itu dilakukan pembuatan *test case erro*r dengan go test dan pengecekan *logging* pada *cloud* Connexi.
- Pembuatan Endpoint untuk *service* reconnect saleschannel pada *orchestrator* baru Connexi. *Reconnect* digunakan untuk penghubungan

ulang *saleschannel*. Terdapat juga pembuatan *test case* dengan go test dan pengujian dengan *postman*.

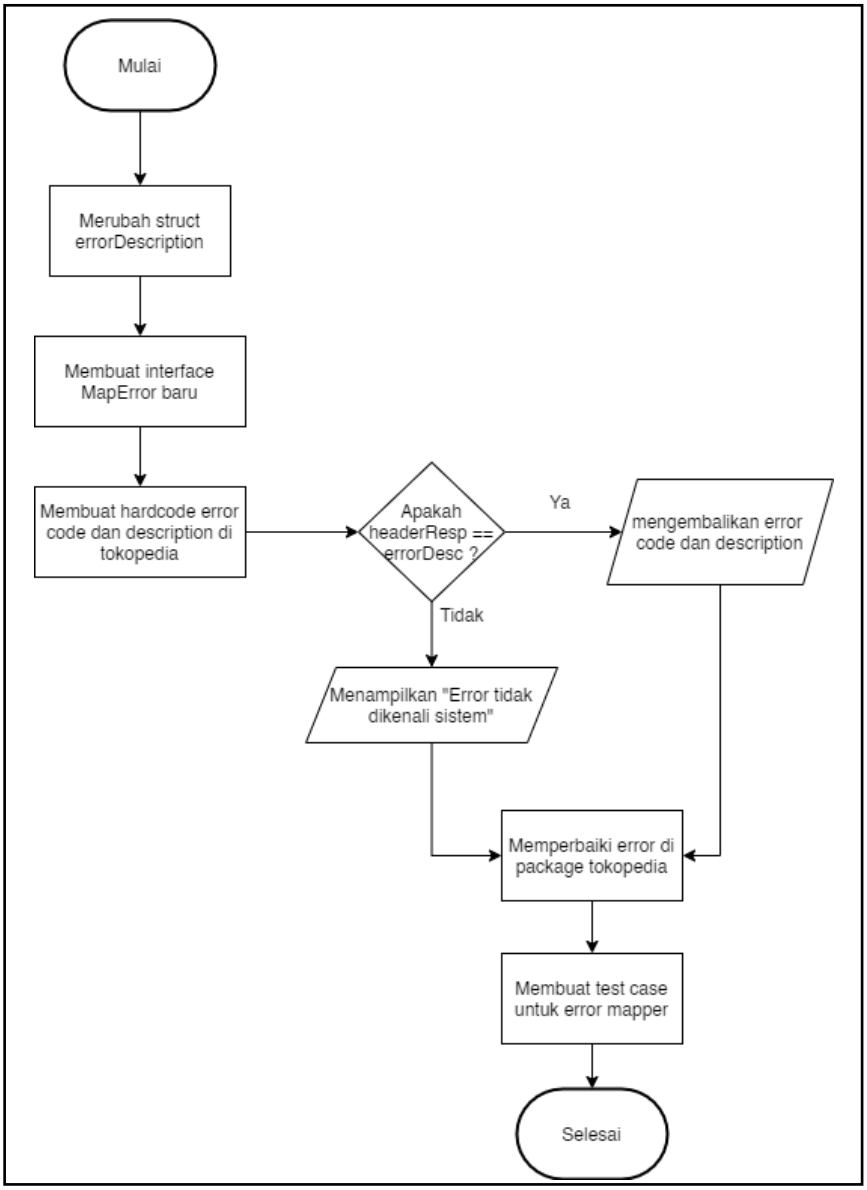
Saat pengerjaan sistem tengah berjalan, pembimbing menambahkan sebuah *requirement* baru untuk sistem error mapping yaitu pengiriman *alert* ke aplikasi Slack Connexi. *Alert* akan dikirim apabila terdapat perbedaan antara hasil *scrapping* dengan *hardcode* yang dibuat di repositori Connexi. Tujuannya agar memudahkan pengecekan pada repositori baru yang dibuat.

### 3.3.3 Perancangan Sistem

#### A. Error Mapping

#### A.1 Flowchart

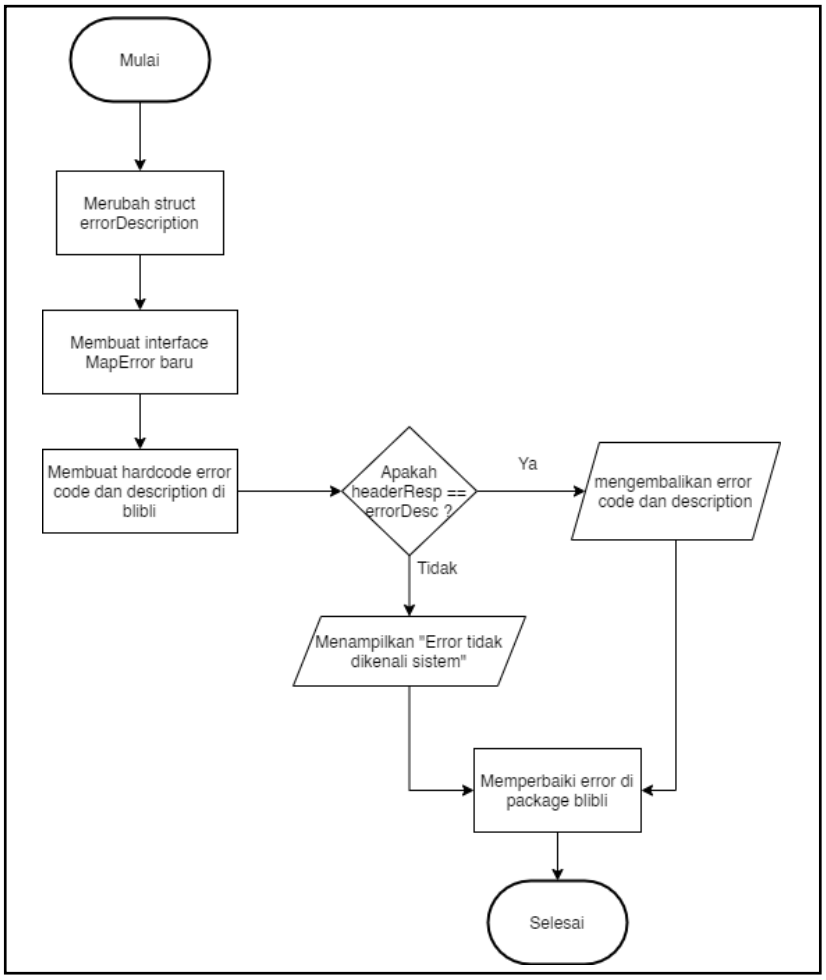
Alur kerja Error Mapping pada Connexi dijelaskan dalam beberapa *flowchart* sebagai berikut.




Gambar 3.1 *Flowchart* Error Mapping Saleschannel Tokopedia

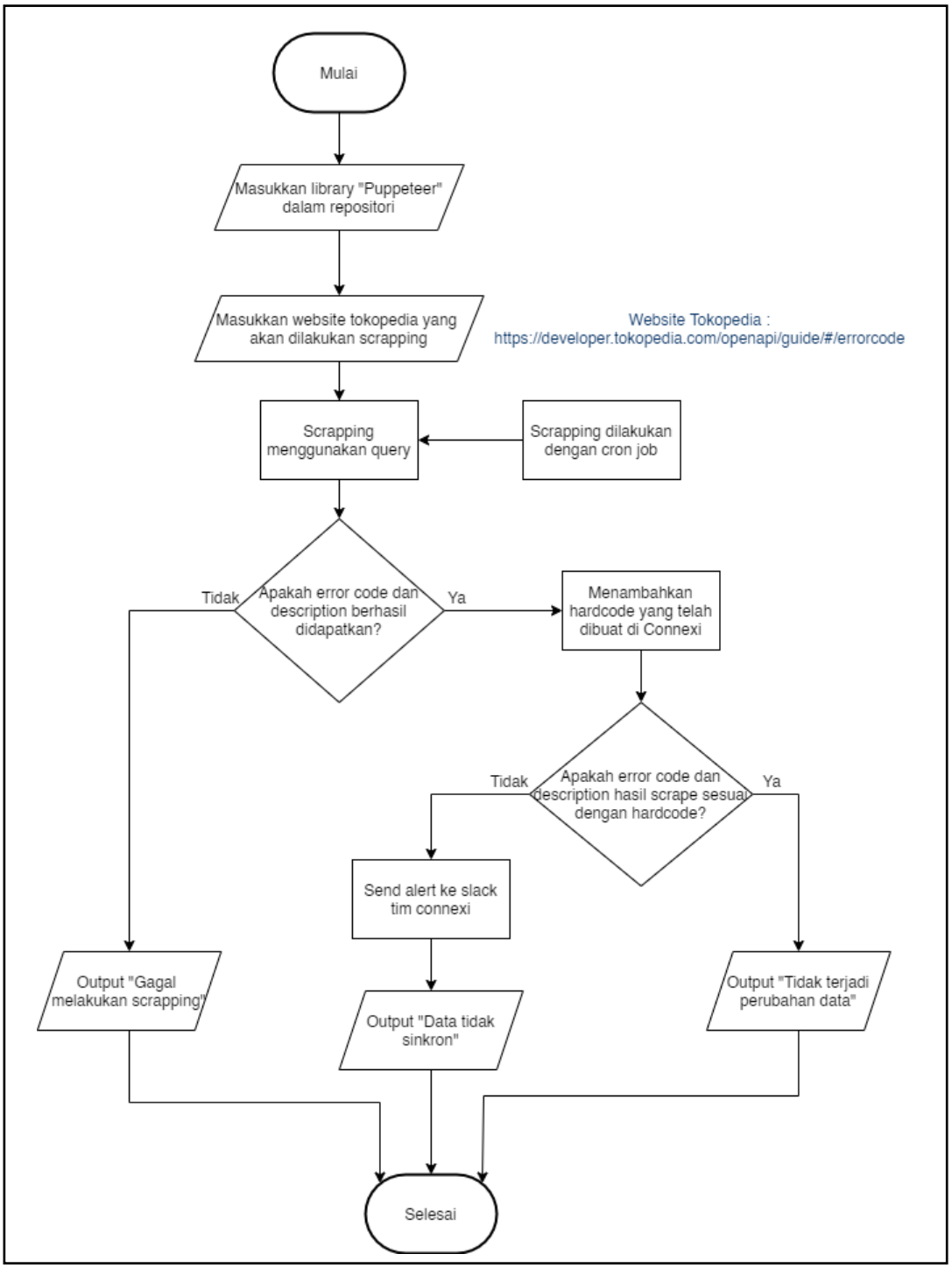
Gambar 3.1 menunjukkan proses pengubahan sistem error mapping pada saleschannel Tokopedia di website Connexi. Akan dibuat *interface* baru bernama *MapError* yang berisi Code dan Deskripsi dari error. Selanjutnya dibuat *hardcode* yang diambil dari website Tokopedia. Alasan digunakannya *hardcode* pada Connexi adalah agar meringankan kerja sistem serta keamanan sistem. Proses

scrapping sendiri akan dilakukan diluar repositori Connexi. Kemudian akan dibuat *test case* pada package Tokopedia menggunakan *go test*.




Gambar 3.2 *Flowchart* Error Mapping Saleschannel Blibli

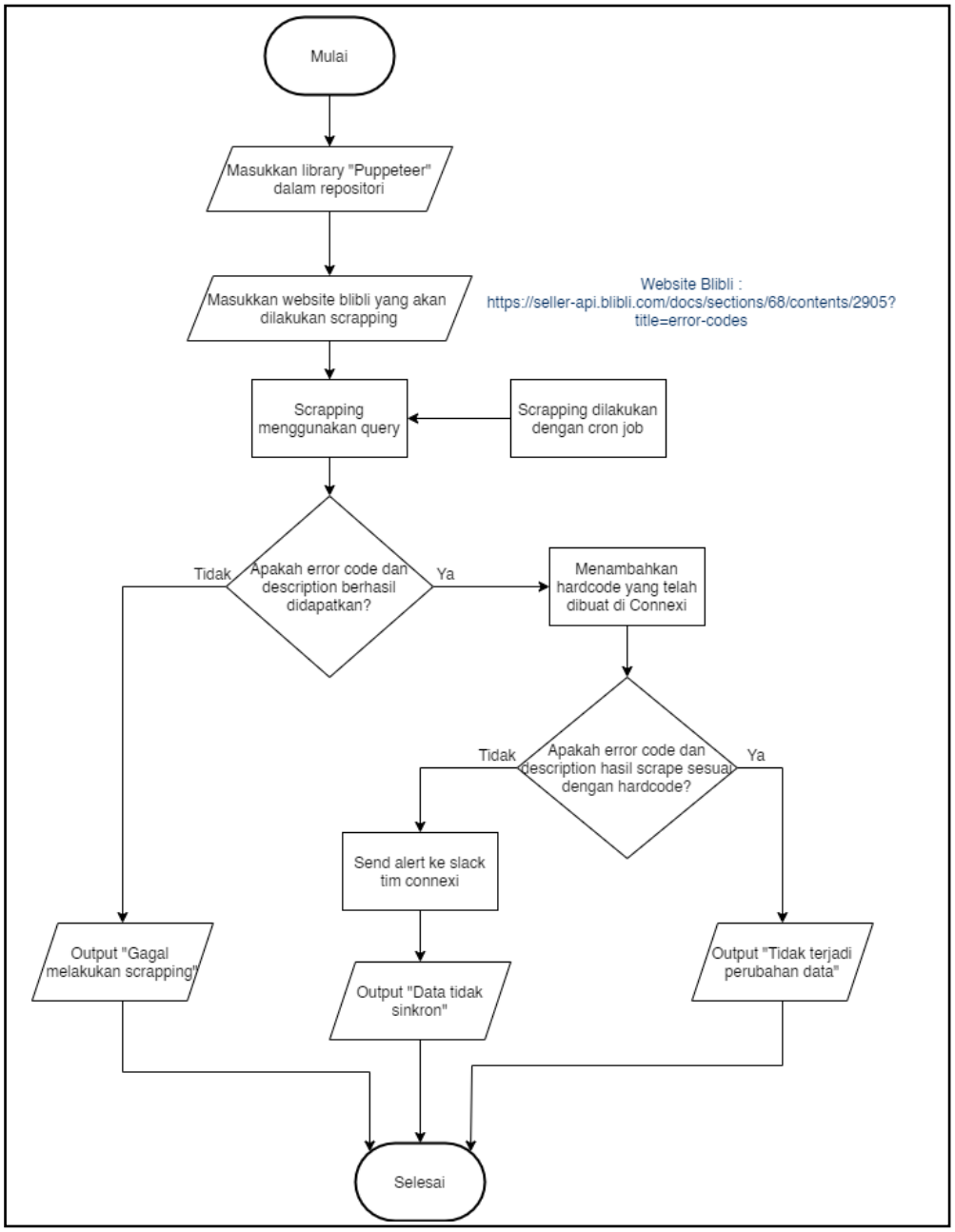
Gambar 3.2 menunjukkan proses pengubahan sistem error mapping pada saleschannel Blibli di website Connexi. Pada saleschannel Blibli juga akan dilakukan *hardcode* yang diambil dari website Blibli.




Gambar 3.3 *Flowchart* Scrapping di Website Tokopedia

Gambar 3.3 menunjukkan proses *scrapping* yang akan dilakukan pada website Tokopedia untuk mengambil code dan deskripsi dari *error*. Proses *scrapping* akan dilakukan dengan node js dan menggunakan library puppeteer.

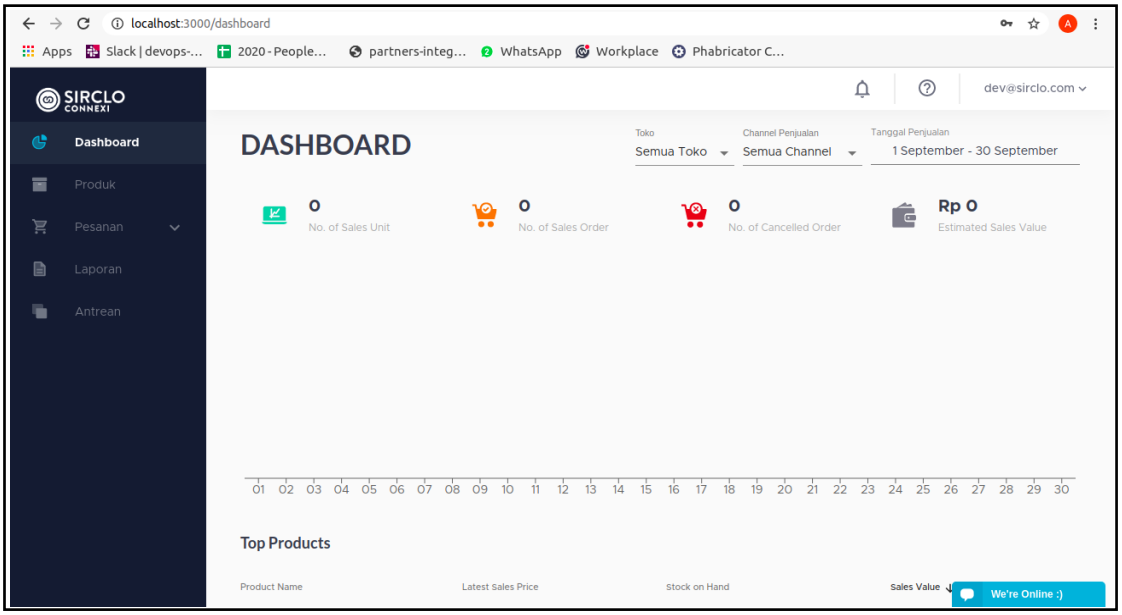
*Scrapping* yang dilakukan juga menggunakan cron job sehingga akan mengotomatisasi *script scrapping* yang dibuat dan dapat dijalankan pada waktu yang telah ditentukan. Pada repositori scrapping ini juga akan dicantumkan *hardcode* yang sebelumnya dibuat di repositori Connexi. Apabila ada perbedaan antara hasil *scrapping* yang didapatkan dengan *hardcode*, maka akan dibuat *alert* yang akan dikirim ke aplikasi *slack* untuk tim Connexi.




Gambar 3.4 *Flowchart* Scrapping di Website Blibli

Gambar 3.4 menunjukkan proses *scrapping* yang akan dilakukan pada website Blibli untuk mengambil code dan deskripsi dari *error*. Proses *scrapping* akan dilakukan dengan node js dan menggunakan library puppeteer. *Scrapping* yang dilakukan juga menggunakan cron job. Pada repositori scrapping ini juga akan dicantumkan *hardcode* yang sebelumnya dibuat di repositori Connexi. Apabila ada perbedaan antara hasil *scrapping* yang didapatkan dengan *hardcode*, maka akan dibuat *alert* yang akan dikirim ke aplikasi *slack* untuk tim Connexi.

## A.2 Implementasi



Gambar 3.5 Halaman Dashboard di Connexi

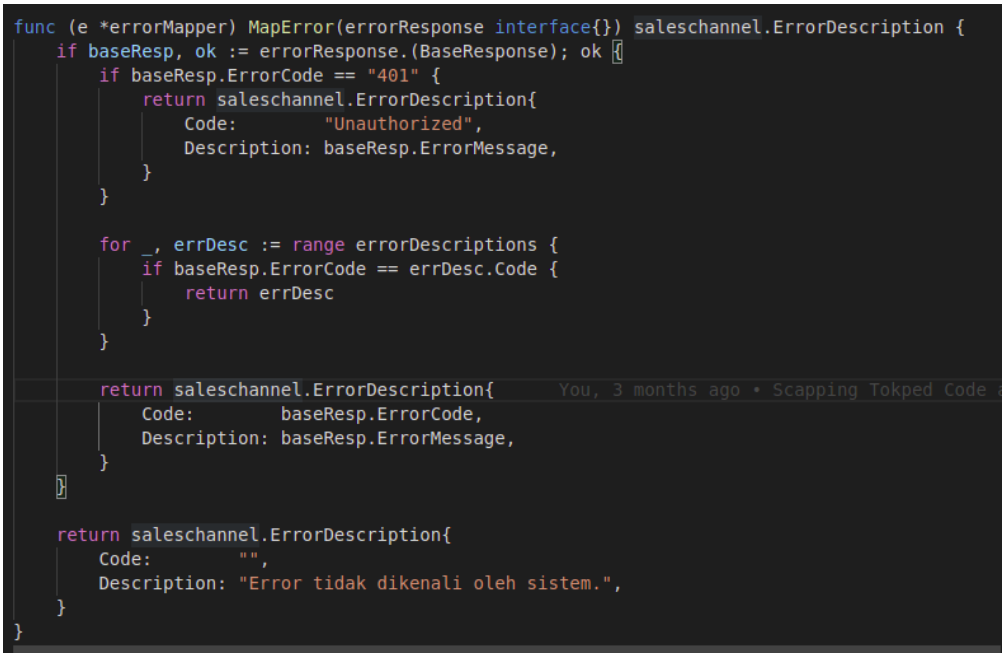
Gambar 3.5 merupakan halaman *dashboard* yang merupakan tampilan pertama yang akan muncul ketika user berhasil login di website Connexi. Setiap data yang digunakan merupakan data dummy. Pemanggilan API dari saleschannel dapat dilakukan di beberapa *page* Connexi, salah satunya halaman *dashboard*.

Setiap user memilih *marketplace* yang akan dilihat maka akan terjadi pemanggilan API terhadap *marketplace* tersebut.

```go
func (e *errorMapper) MapError(errorResponse interface{}) saleschannel.ErrorDescription {
    if headerResp, ok := errorResponse.(ResponseHeader); ok {      You, 3 months ago • Scapping
        for _, errDesc := range errorDescriptions {
            if headerResp.ErrorCode == errDesc.Code {
                return errDesc
            }
        }

        return saleschannel.ErrorDescription{
            Code:        headerResp.ErrorCode,
            Description: headerResp.Reason,
        }
    }

    return saleschannel.ErrorDescription{
        Code:        "",
        Description: "Error tidak dikenali oleh sistem.",
    }
}
```

Gambar 3.6 Potongan *code function* MapError Tokopedia di Connexi

Gambar 3.6 merupakan potongan *code* pada *package* Tokopedia di web Connexi. Pada *function* ini akan dilakukan pengecekan antara respon error yang didapat saat pemanggilan API dengan variabel errorDescription yang berisi *hardcode* dari web Tokopedia. Apabila data sama maka *log* akan berisi kode dan deskripsi error yang bersangkutan. Sementara jika berbeda, maka akan dimunculkan *log* yang hanya berisi deskripsi error bahwa error tidak dikenali sistem.



```
func (e *errorMapper) MapError(errorResponse interface{}) saleschannel.ErrorDescription {
    if baseResp, ok := errorResponse.(BaseResponse); ok {
        if baseResp.ErrorCode == "401" {
            return saleschannel.ErrorDescription{
                Code:        "Unauthorized",
                Description: baseResp.ErrorMessage,
            }
        }

        for _, errDesc := range errorDescriptions {
            if baseResp.ErrorCode == errDesc.Code {
                return errDesc
            }
        }

        return saleschannel.ErrorDescription{      You, 3 months ago • Scapping Tokped Code
            Code:        baseResp.ErrorCode,
            Description: baseResp.ErrorMessage,
        }
    }

    return saleschannel.ErrorDescription{
        Code:        "",
        Description: "Error tidak dikenali oleh sistem.",
    }
}
```

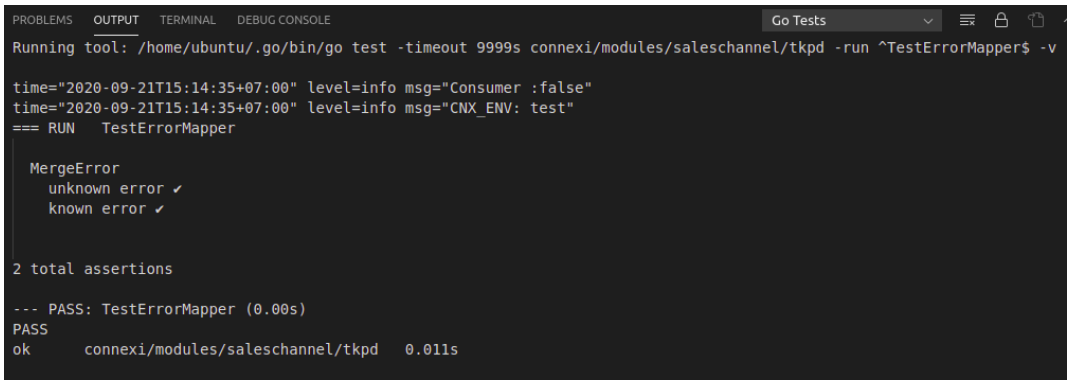
Gambar 3.7 Potongan *code function* MapError Blibli di Connexi

Gambar 3.7 merupakan potongan *code* pada *package* Blibli di web Connexi. Pada *function* ini juga akan dilakukan pengecekan antara respon error yang didapat saat pemanggilan API dengan variabel errorDescription yang berisi *hardcode* dari web Blibli. Apabila data sama maka *log* akan berisi kode dan deskripsi error yang bersangkutan. Sementara jika berbeda, maka akan dimunculkan *log* yang hanya berisi deskripsi error bahwa error tidak dikenali sistem. Pada MapError di Blibli ini terdapat penambahan pengecekan kode error, dimana apabila kode error yang didapat berupa 401 maka akan dimunculkan *log* berupa "*Unathorized Access*".

1. Pengujian dengan go test

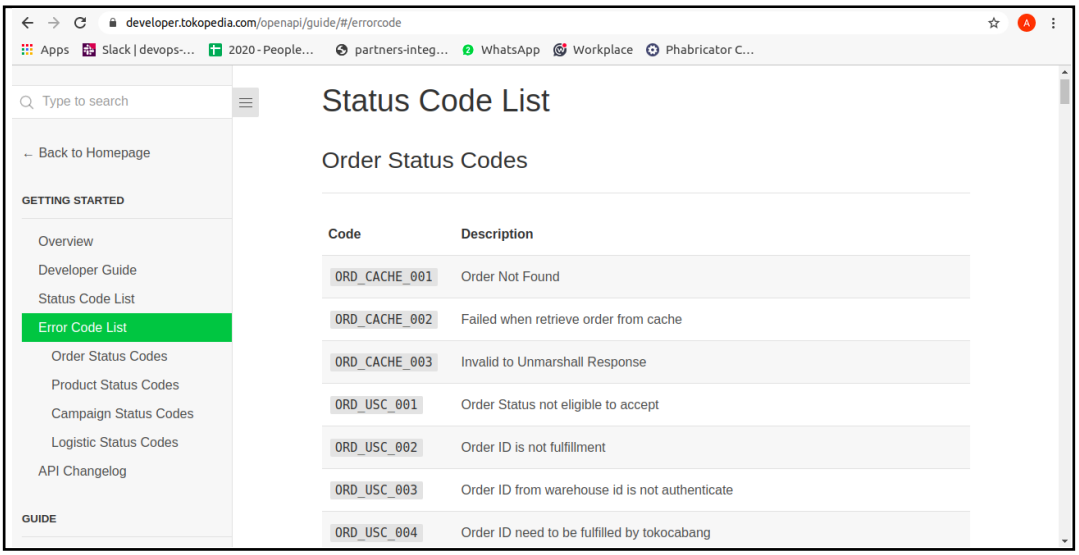
Pengujian ini dilakukan pada repositori website Connexi dengan menggunakan *package* khusus yang disediakan golang. *Package* ini bernama go test yang digunakan untuk melakukan *automated testing*. Proses *testing* dilakukan untuk mengecek setiap pemanggilan *function* yang ada di error_mapper.go. Akan dibuat

sebuah file baru bernama error_mapper_test.go untuk melakukan *automated testing* ini, pengujian dilakukan dengan menggunakan data dummy.

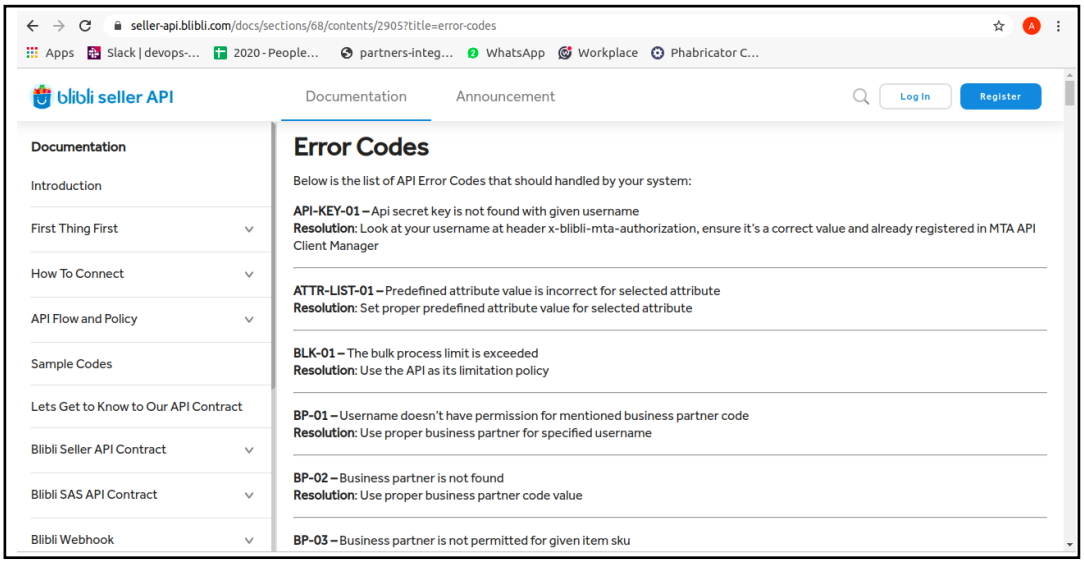



Gambar 3.8 Hasil Pengujian dengan Go Test

2. Pengambilan Data dan Pengujian Hasil Scrapping pada Node JS




Gambar 3.9 Tampilan Error List pada Website Tokopedia




Gambar 3.10 Tampilan Error List pada Website Blibli

Gambar 3.9 dan 3.10 merupakan tampilan *error list* pada website Tokopedia dan Blibli yang akan dilakukan *scrapping*. *Scrapping* akan dilakukan untuk mendapatkan data berupa *code* dan deskripsi error.

```
const puppeteer = require('puppeteer');
const { format } = require('path');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();

  let tkpd_toko_official = 'https://developer.tokopedia.com/openapi/guide/#/errorcode';
    await page.goto(tkpd_toko_official,{ waitUntil:'networkidle0', timeout: 0});

    let data = await page.evaluate(()=>{

        let size = document.querySelectorAll('.table-wrapper table tbody tr').length;
        let results = [];
        for(i = 0 ; i < size; i++){
            results.push({
                Code : document.querySelectorAll('code')[i].textContent,
                Description : document.querySelectorAll('td:nth-child(2)')[i].textContent
            });
        }
    });

  await browser.close();

  console.log(data);
  return data;
})();
```

Gambar 3.11 Potongan *code scrapping* web Tokopedia

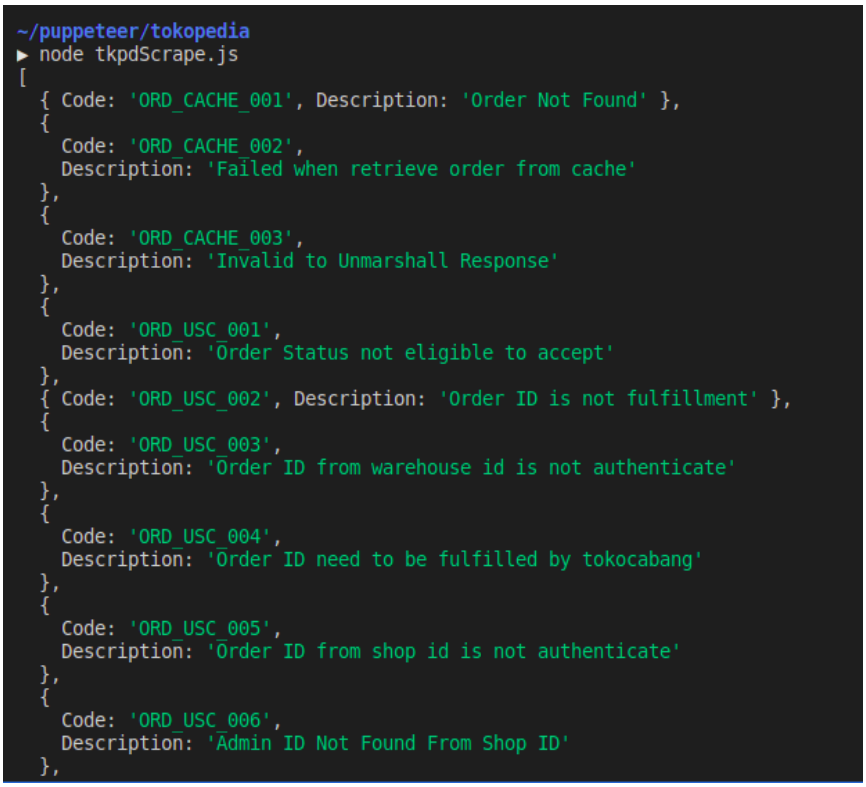
```
const puppeteer = require('puppeteer');
const { format } = require('path');

(async () => {
  const browser = await puppeteer.launch();
  const page = await browser.newPage();

  let blib_toko_official = 'https://seller-api.blibli.com/docs/sections/68/contents/2905?title=error-codes';
    await page.goto(blib_toko_official,{ waitUntil:'networkidle0', timeout: 0});

    let data = await page.evaluate(()=>{
        let size = document.querySelectorAll(".content-body p").length;
        let results = [];
        for(i = 0 ; i < size; i++){
            results.push({
                Code : document.querySelectorAll('strong:nth-child(1)')[i].textContent,
                Desc : document.querySelectorAll('strong:nth-child(2)')[i].textContent
            });
        }
    });

  await browser.close();

  console.log(data);
  return data;
})();
```

Gambar 3.12 Potongan *code scrapping* web Blibli

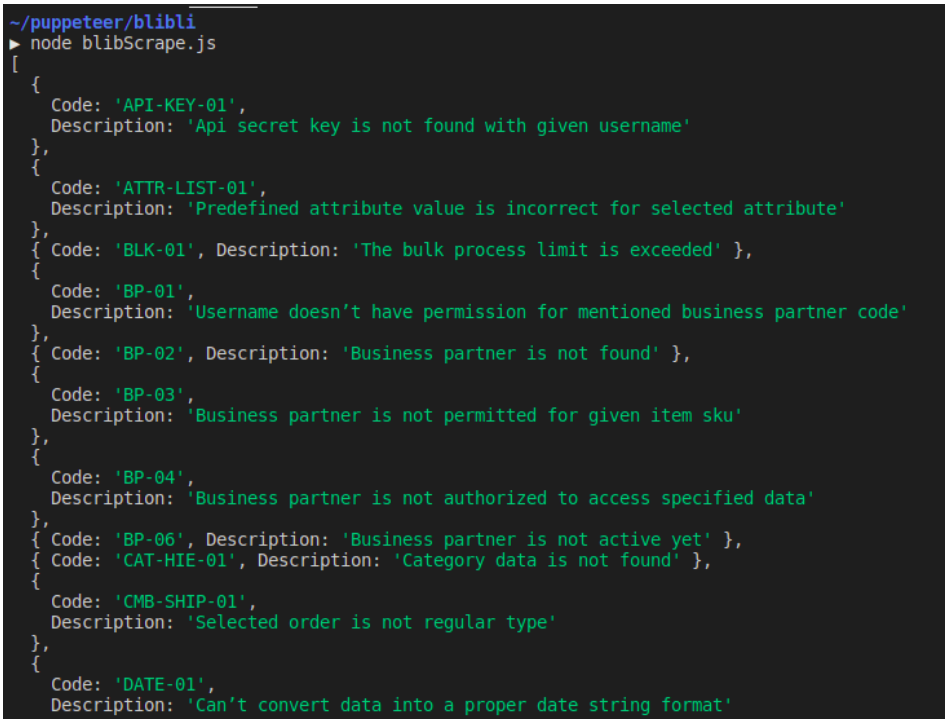
Gambar 3.11 dan 3.12 merupakan potongan *code* proses *scrapping* pada web Tokopedia dan Blibli. *Scrapping* dilakukan dengan membuat *query* untuk mengambil data kode dan deskripsi error. Proses ini dilakukan di dalam repositori baru yang ditujukan untuk melakukan *scrapping* dengan Node JS. Proses *scrapping* akan menampilkan *code* dan deskripsi error yang didapatkan dari kedua web saleschannel. Proses *scrapping* dilakukan menggunakan *library puppeteer* serta digunakan *cron job* dengan konfigurasi pengecekan setiap hari pada tengah malam.

```
var request = require('request');
    cheerio = require('cheerio');
    util = require('util');
    http = require('http');
    cronJob = require('cron').CronJob;

var data = new Array();

new cronJob('0 0 * * *', function(){
```

Gambar 3.13 Konfigurasi Cron Job untuk *Scrapping*




Gambar 3.14 Hasil *Scrapping* Web Tokopedia

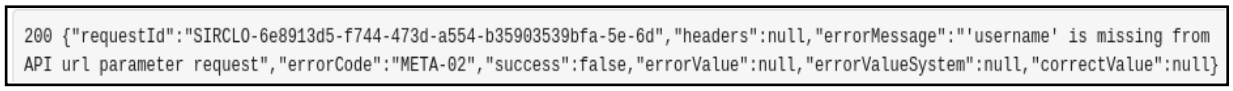



Gambar 3.15 Hasil *Scrapping* Web Blibli

Hasil *scrapping* dari web Tokopedia dan Blibli ditunjukkan pada gambar 3.14 dan 3.15. Dari pengecekan yang dilakukan, kode dan deskripsi error yang didapatkan telah sesuai dengan data pada web Tokopedia dan Blibli.

3. Pengujian Logging pada Google Cloud

Pengecekan Logging dilakukan pada Cloud Connexi untuk melihat respon yang dihasilkan saat pemanggilan API. Akan dilakukan request ke saleschannel, kemudian hasil log yang didapat akan menampilkan errorMessage serta errorCode.

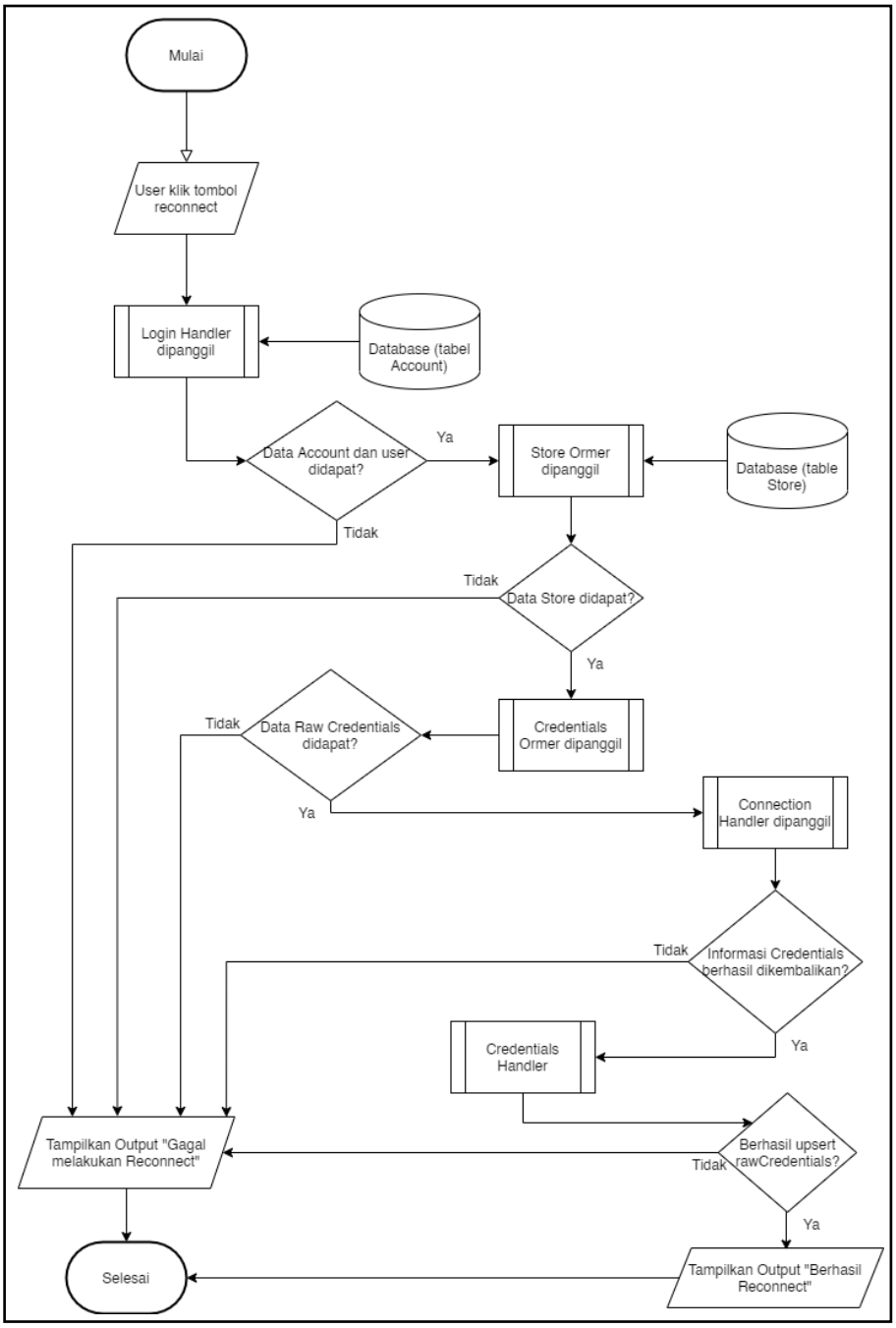



Gambar 3.16 Tampilan Log Error pada Cloud Connexi

## B. Reconnect Saleschannel

### B.1 Flowchart dan UML Diagram

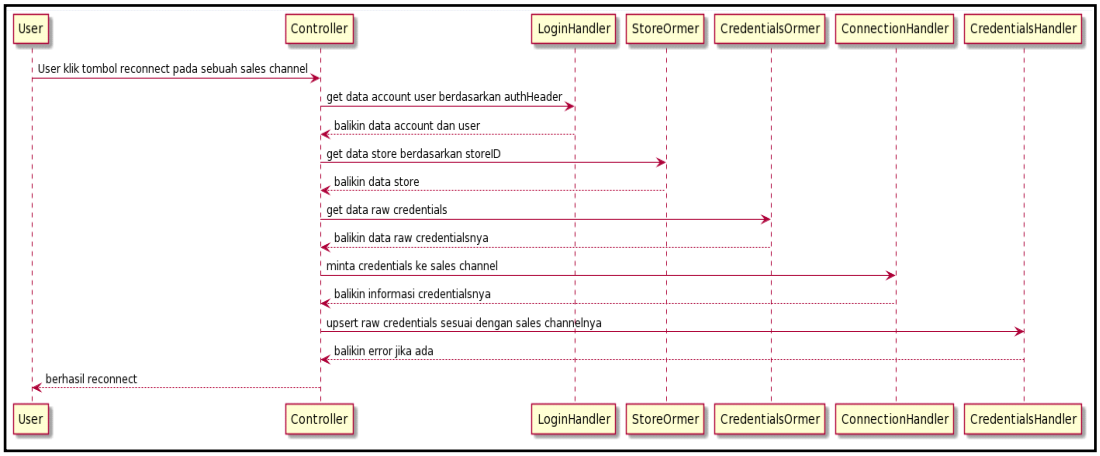
Alur kerja Reconnect Saleschannel pada Connexi dijelaskan dalam sebuah *flowchart* dan diagram UML sebagai berikut.




Gambar 3.17 *Flowchart* Reconnect Saleschannel

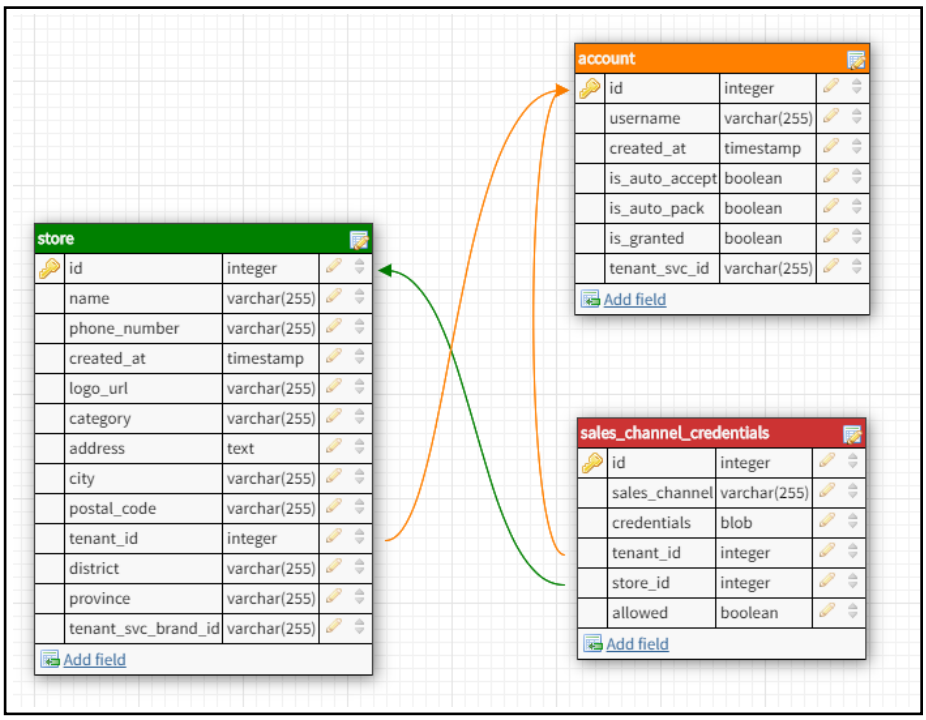
Gambar 3.17 merupakan *flowchart* Reconnect saleschannel untuk *orchestrator* baru Connexi. Terdapat lima *handler* yang akan dipanggil ketika user melakukan reconnect saleschannel. Login *handler* dan Store ormer akan mengambil data account dan store dari user yang bersangkutan. Credentials ormer

kemudian mengambil data *raw credentials* user. Setelah itu ada Connection *handler* yang menangani koneksi ulang saleschannel. Terakhir terdapat Credentials *handler* untuk melakukan *update raw credentials* baru user. Ketika semua *handler* tersebut berhasil dipanggil maka proses reconnect saleschannel berhasil dilakukan.




Gambar 3.18 UML Diagram Reconnect Saleshannel

## B.2 Relasi Antar Tabel




Gambar 3.19 Relasi Antar Tabel untuk Reconnect Saleschannel

Gambar 3.19 menunjukkan tiga tabel yang utama yang digunakan dalam melakukan proses reconnect saleschannel. Tabel account digunakan untuk melakukan pengambilan data user dari tenant_id di tabel sales_channel_credentials, tabel store digunakan untuk pengambilan data store berdasarkan store_id yang ada di tabel sales_channel_credentials. Tabel sales_channel_credentials digunakan untuk pengambilan credentials saat proses reconnect dan melakukan *update* pada *field* allowed.

### B.3 Struktur Tabel

Database yang digunakan adalah PostgreSQL. Berikut adalah struktur tabel yang digunakan dalam melakukan reconnect saleschannel.

Tabel 3.2 Tabel store

| Nama Kolom | Tipe Data | Panjang Data |
|---|---|---|
| id (PK) | integer | |
| Name | varchar | 255 |
| phone_number | varchar | 255 |
| created_at | timestamp | |
| logo_url | varchar | 255 |
| Category | varchar | 255 |
| Address | text | |
| City | varchar | 255 |
| postal_code | varchar | 255 |
| tenant_id (FK) | integer | |
| District | varchar | 255 |
| Province | varchar | 255 |
| tenant_svc_brand_id | varchar | 255 |

Tabel 3.2 merupakan struktur dari tabel store pada database yang digunakan. Tabel store digunakan untuk menyimpan data – data setiap store yang ada di Connexi.

Tabel 3.3 Tabel sales_channel_credentials

| Nama Kolom | Tipe Data | Panjang Data |
|---|---|---|
| id (PK) | integer | |
| sales_channel | varchar | 255 |
| credentials | jsonb | |
| tenant_id (FK) | integer | |
| store_id (FK) | integer | |
| allowed | boolean | |

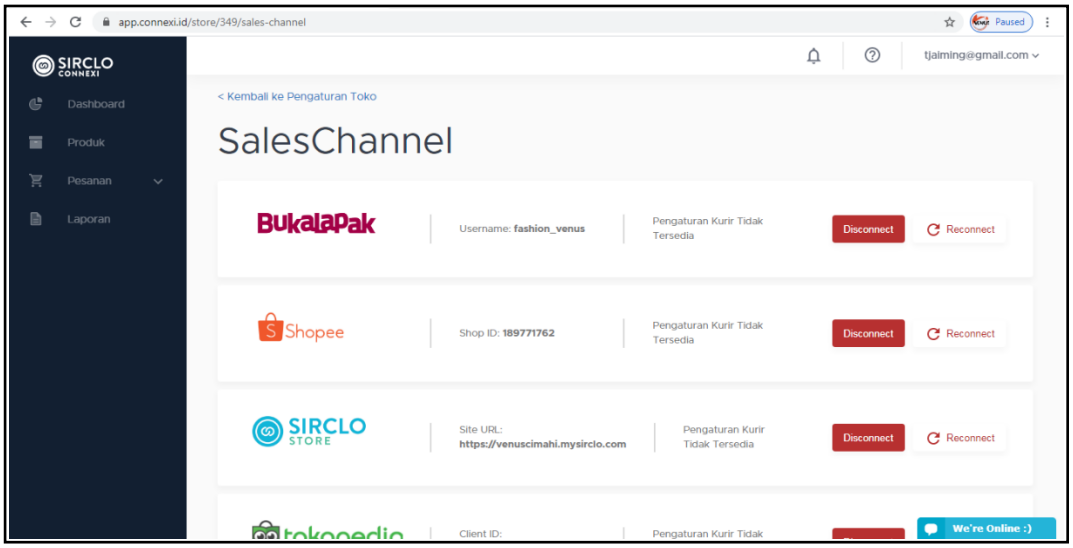
Tabel 3.3 merupakan struktur dari tabel sales_channel_credentials pada database yang digunakan. Tabel ini digunakan untuk menyimpan data credentials setiap saleschannel yang ada di Connexi.

Tabel 3.4 Tabel account

| Nama Kolom | Tipe Data | Panjang Data |
|---|---|---|
| id (PK) | integer | |
| username | varchar | 255 |
| created_at | timestamp | |
| is_auto_accept | boolean | |
| is_auto_pack | boolean | |
| is_granted | boolean | |
| tenant_svc_id | varchar | 255 |

Tabel 3.4 merupakan struktur dari tabel account pada database yang digunakan. Tabel ini digunakan untuk menyimpan data semua user yang ada di Connexi.

## B.4 Implementasi




Gambar 3.20 Reconnect Saleschannel pada Web Connexi

Gambar 3.20 menunjukkan tampilan tombol reconnect pada website Connexi. Tampilan ini akan muncul saat *user* mengklik pengaturan toko. Saat *user* melakukan klik terhadap tombol reconnect maka *function* reconnect pada *package* saleschannel di repositori *backend* akan dipanggil.

```go
func (c *channelService) reconnect(ctx context.Context, channelCode string, brandID string) error {
    constructor, err := c.channelRegistry.GetConstructor(channelCode)
    if err != nil {
        log.WithContext(ctx).Error(stacktrace.Propagate(err, "[GetConstructor] Channel registry error for channel %s", chanr
        return err
    }

    store, err := c.storeOrmer.GetByTenantServiceBrandID(brandID)
    if err != nil {
        log.WithContext(ctx).Error(stacktrace.Propagate(err, "[GetByTenantServiceBrandID] Store ormer error for brand id %s'
        return err
    }

    rawCredentials, err := c.credentialsOrmer.Get(channelCode, store.TenantID, "")
    if err != nil {
        log.WithContext(ctx).Error(stacktrace.Propagate(err, "[Get] Credentials ormer error for tenant %d with channel %s",
        return err
    }

    // change the credential data and re-mapping data json
    var iCredential map[string]interface{}

    err = json.Unmarshal([]byte(rawCredentials.Credentials), &iCredential)
    if err != nil {
        log.WithContext(ctx).Error(stacktrace.Propagate(err, "[Unmarshal] JSON error"))
        return err
    }
```

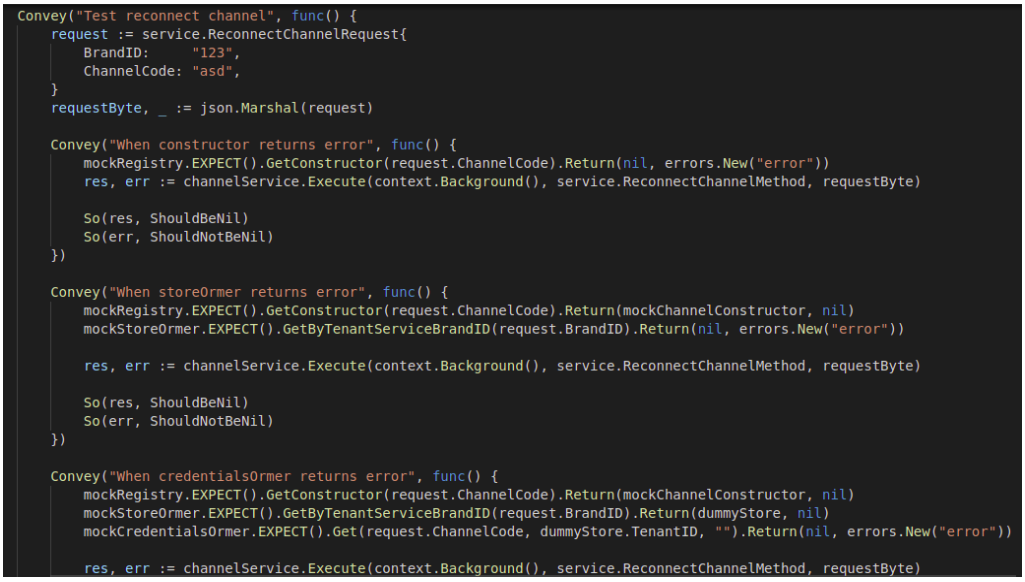
Gambar 3.21 Potongan *Code Function* Reconnect pada Web Connexi

Gambar 3.21 merupakan potongan *code* dari *function* reconnect. Akan ada lima *handler* yang dipanggil ketika user melakukan reconnect saleschannel. Pertama akan dipanggil Login *handler* dan Store ormer yang akan mengambil data account dan store dari user yang bersangkutan. Kemudian Credentials ormer akan mengambil data *raw credentials* user. Selanjutnya ada Connection *handler* yang menangani koneksi ulang saleschannel. Terakhir terdapat Credentials *handler* untuk melakukan *update raw credentials* baru user. Ketika semua *handler* tersebut berhasil dipanggil maka proses reconnect saleschannel berhasil dilakukan.

Pada bagian *backend*, terdapat dua pengujian yang dilakukan untuk memastikan proses reconnect ini berjalan sesuai ketentuan. Kedua pengujian tersebut dijabarkan sebagai berikut.

1. Pengujian menggunakan go test

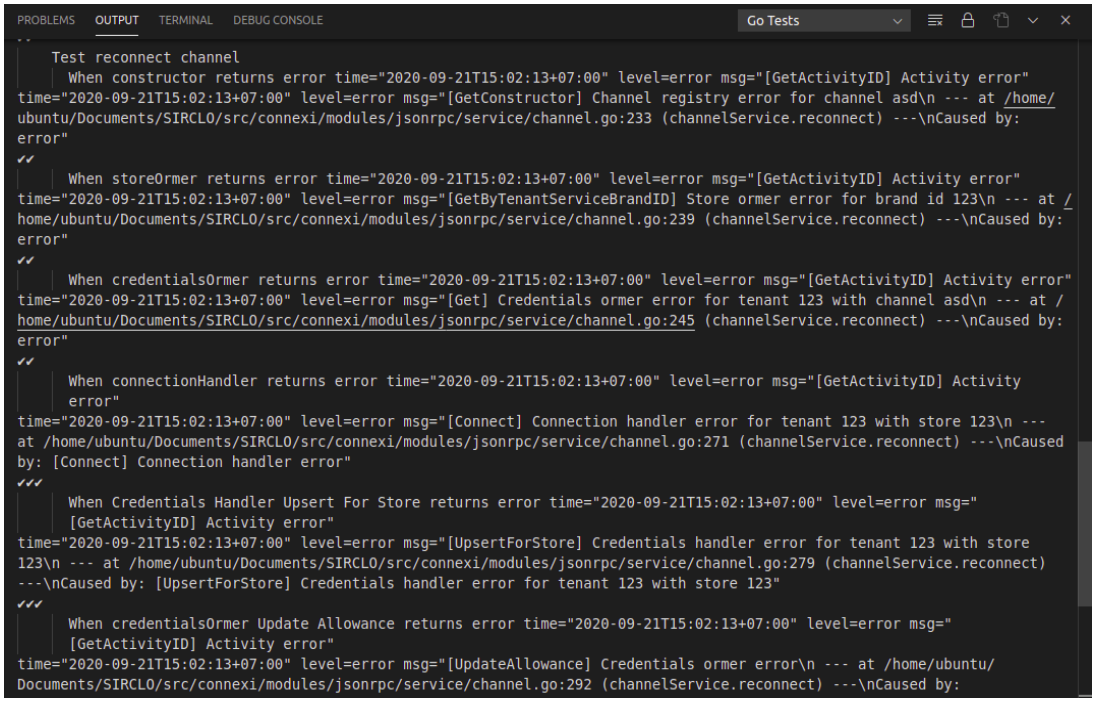
Pada bahasa pemrograman golang, terdapat *package* khusus yang digunakan untuk melakukan *testing*. *Package* tersebut yaitu go test yang digunakan untuk melakukan *automated testing*. *Command* ini akan mengeksekusi fungsi pengujian apapun yang memiliki *extension* " *_test.go ". Misalnya dalam pengujian ini, proses reconnect ditempatkan pada file client.go, maka untuk pengujiannya akan dibuat file bernama client_test.go yang ada di satu *package* yang sama.



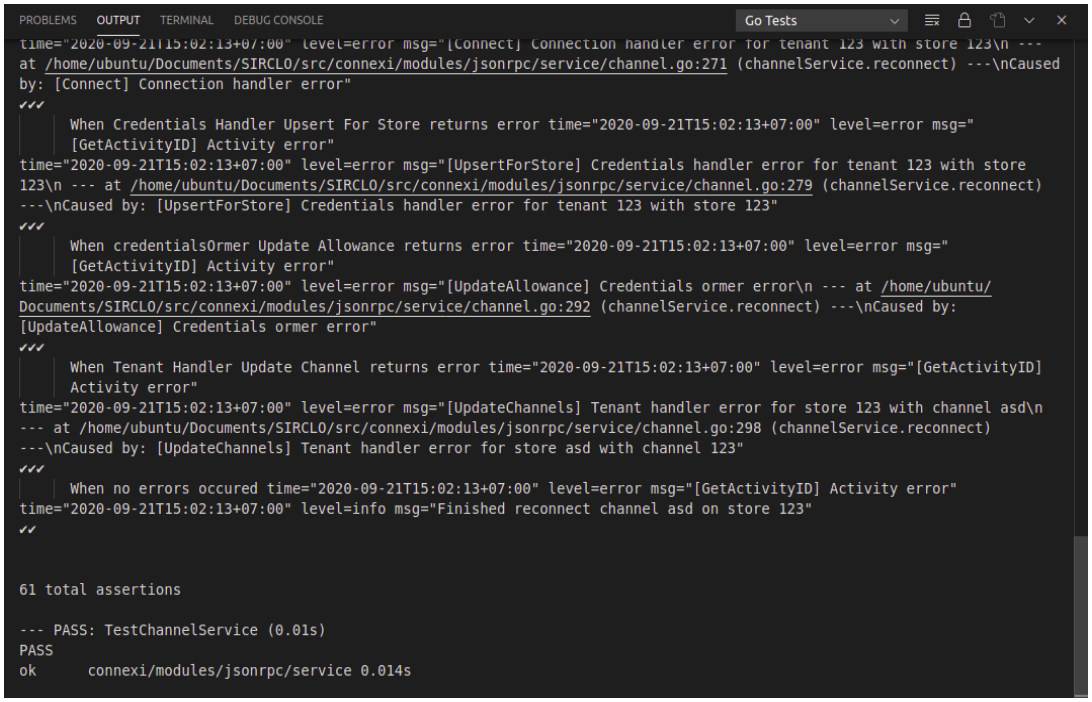
```
Convey("Test reconnect channel", func() {
    request := service.ReconnectChannelRequest{
        BrandID:     "123",
        ChannelCode: "asd",
    }
    requestByte, _ := json.Marshal(request)

    Convey("When constructor returns error", func() {
        mockRegistry.EXPECT().GetConstructor(request.ChannelCode).Return(nil, errors.New("error"))
        res, err := channelService.Execute(context.Background(), service.ReconnectChannelMethod, requestByte)

        So(res, ShouldBeNil)
        So(err, ShouldNotBeNil)
    })

    Convey("When storeOrmer returns error", func() {
        mockRegistry.EXPECT().GetConstructor(request.ChannelCode).Return(mockChannelConstructor, nil)
        mockStoreOrmer.EXPECT().GetByTenantServiceBrandID(request.BrandID).Return(nil, errors.New("error"))

        res, err := channelService.Execute(context.Background(), service.ReconnectChannelMethod, requestByte)

        So(res, ShouldBeNil)
        So(err, ShouldNotBeNil)
    })

    Convey("When credentialsOrmer returns error", func() {
        mockRegistry.EXPECT().GetConstructor(request.ChannelCode).Return(mockChannelConstructor, nil)
        mockStoreOrmer.EXPECT().GetByTenantServiceBrandID(request.BrandID).Return(dummyStore, nil)
        mockCredentialsOrmer.EXPECT().Get(request.ChannelCode, dummyStore.TenantID, "").Return(nil, errors.New("error"))

        res, err := channelService.Execute(context.Background(), service.ReconnectChannelMethod, requestByte)
```

Gambar 3.22 Potongan *Code Test Case* untuk Reconnect Saleschannel

Proses *testing* dilakukan dengan mengecek setiap pemanggilan *function* yang ada di client.go. Pada client_test.go akan dibuat sebuah *mock* data yang berisikan brand_id dan channel_code untuk melakukan proses reconnect. *Testing* dibutuhkan untuk mengetahui respon yang diberikan *server* terhadap masing-masing *function* saat berhasil atau terjadi kegagalan dalam pemanggilannya. Hasil pengujian reconnect saleschannel dengan go test dapat dilihat pada gambar 3.23 dan 3.24. *Unit test* pada setiap *function* berhasil dilakukan dan memunculkan hasil *log* sesuai kondisi yang diinginkan.



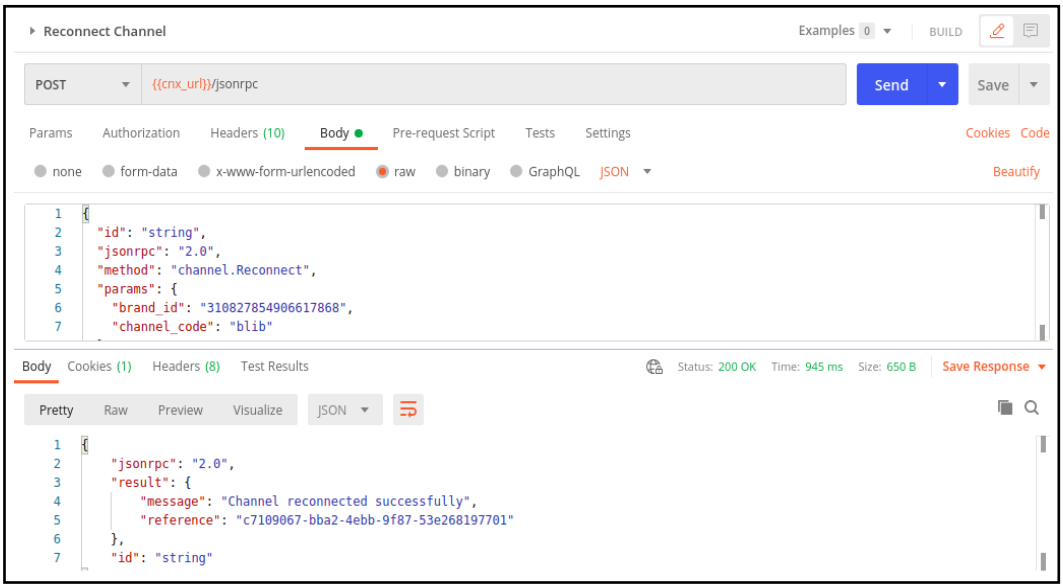

Gambar 3.23 Pengujian Reconnect Saleschannel dengan Go Test




Gambar 3.24 Pengujian Reconnect Saleschannel dengan Go Test (Lanjutan)

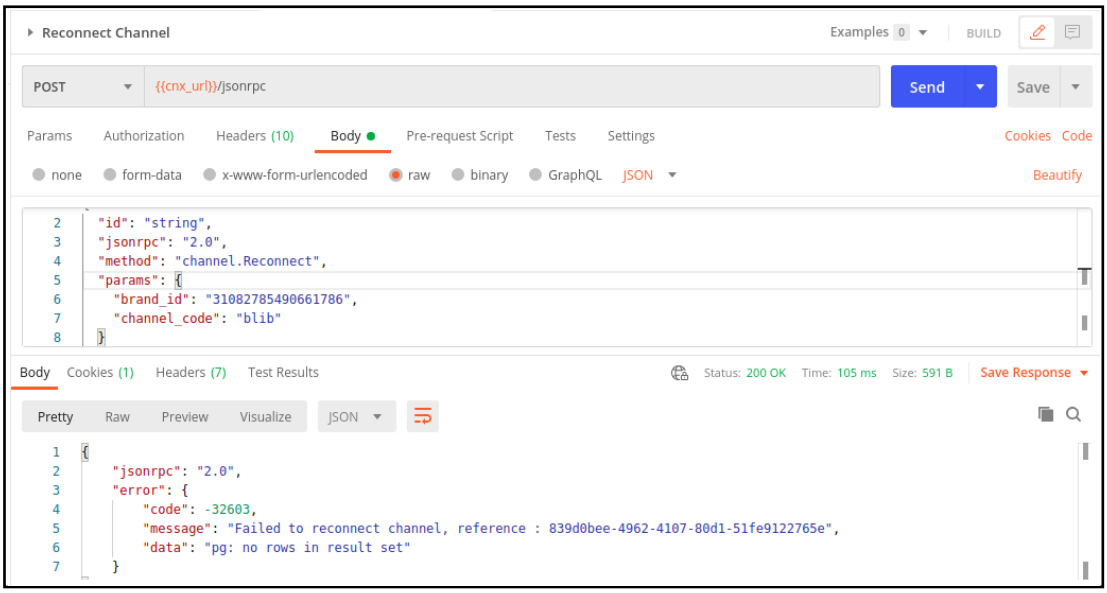
2. Pengujian menggunakan Postman

Postman adalah tool yang digunakan untuk uji coba REST API. Pada postman, terlebih dahulu akan dibuat data / *resource* baru dengan menggunakan metode POST. Terdapat brand_id dan channel_code yang dijadikan sebagai parameter. Proses pengujian reconnect saleschannel dengan postman dapat dilihat pada gambar 3.25 dan 3.26.




Gambar 3.25 Pengujian Reconnect Saleschannel dengan Postman (Berhasil)

Gambar 3.25 menunjukkan kondisi apabila proses reconnect saleschannel berhasil dilakukan. Akan muncul pesan yang berisikan informasi bahwa channel berhasil melakukan reconnect. Pengujian dengan postman ini dilakukan dengan pemilihan *staging* sebagai *environment*. Parameter yang dimasukkan berupa brand_id dan channel_code sudah terdapat di *staging* Connexi sebelumnya.




Gambar 3.26 Pengujian Reconnect Saleschannel dengan Postman (Gagal)

Gambar 3.26 menunjukkan kondisi apabila proses reconnect saleschannel gagal dilakukan. Akan muncul pesan yang berisikan informasi error yang berisikan pesan bahwa channel gagal melakukan reconnect, kode error dan detailnya. Pengujian dengan postman ini dilakukan dengan pemilihan *staging* sebagai *environment*. Sebelumnya, parameter yang dimasukkan berupa brand_id dan channel_code tidak terdapat di *staging* Connexi sehingga menampilkan error pada postman.

### 3.3.4 Kendala yang Ditemukan

Kendala yang ditemukan selama pelaksanaan magang di PT Lingkar Niaga Solusindo antara lain :

- Diperlukannya waktu bagi penulis dalam beradaptasi pada *environment* Connexi, seperti bahasa pemrograman golang, penggunaan Arcanist dan Docker.

- Saat memasuki *staging,* terkadang beberapa task ada yang belum sesuai *requirement* atau memerlukan *requirement* baru. Hal ini terkadang membuat waktu pengerjaan *task* menjadi lebih lama.
- Pengerjaan *unit test* untuk setiap komponen pada *task* juga terkadang bermasalah, karena belum adanya alur dalam melakukan *unit test* sehingga kadang membingungkan *developer*.

### 3.3.5 Solusi Atas Kendala yang Ditemukan

Setiap kendala yang ditemukan pasti memiliki cara penyelesaiannya masing-masing. Beberapa solusi yang digunakan untuk mengatasi kendala-kendala tersebut antara lain :

- Selalu berkomunikasi dengan pembimbing untuk setiap *task* yang akan dikerjakan.
- Mengadakan *Backlog Refinement* (BR) lebih lama setiap minggunya agar *task* yang dikerjakan menjadi lebih jelas dan terarah.
- Selalu mengikuti *daily stand-up* untuk membahas *progress* setiap *task* yang dikerjakan maupun permasalahan yang dihadapi saat mengerjakan *task* tersebut.