PETUNJUK OPERASIONAL SISTEM KOMPUTER - WS PBB

20 Januari 2017

Priyanto Tamami, S.Kom.

1 PENDAHULUAN

Sistem aplikasi ini digunakan untuk melakukan komunikasi dengan pihak Bank sebagai tempat pembayaran yang difungsikan untuk mencatatkan pembayaran Pajak Bumi dan Bangunan Perdesaan dan Perkotaan.

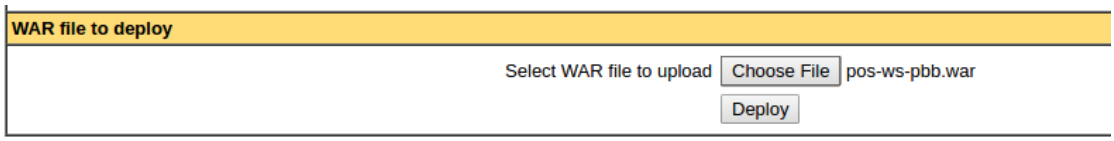
Bentuk dari aplikasi ini hanya menyediakan *Application Programming Interface* (API) untuk melakukan *inquiry* data tagihan Pajak Bumi dan Bangunan Perdesaan dan Perkotaan, atau melakukan pencatatan pembayaran yang terjadi, atau melakukan *reversal* bila terjadi kesalahan pencatatan pembayaran.

Karena implementasinya berbentuk REST API, maka jalur komunikasi yang digunakan sama dengan jalur komunikasi aplikasi *web* pada umumnya, yaitu menggunakan *port* 80.

2 INSTALASI PROGRAM

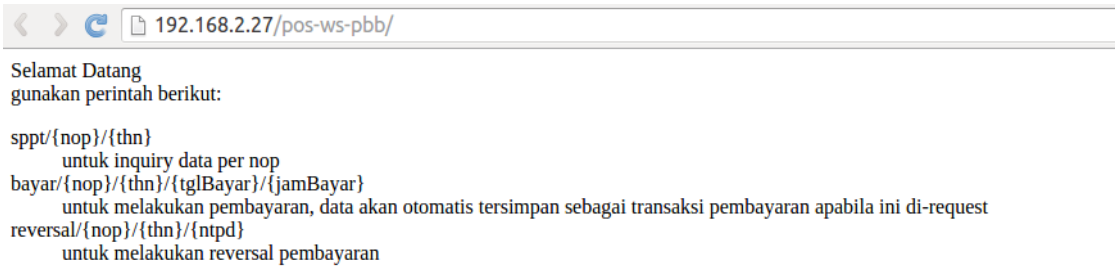
Instalasi program aplikasi *web service* ini cukup mudah, karena hasil dari kompilasi kode ke bentuk program akan berbentuk sebuah *file* berekstensi `war`, yang nantinya *file* ini akan diunggah ke *server* Tomcat untuk melayani *browser* di masing-masing *client*. Langkah instalasi yang dilakukan adalah sebagai berikut :

1. Melakukan kompilasi kode yang menghasilkan *file* berekstensi `war`.
2. Melakukan unggah *file* `war` ke *server* Tomcat seperti pada gambar 1 berikut :



Gambar 1: *Deploy File* `war`

3. Melakukan pengujian dengan melakukan akses ke *server* menggunakan *browser*, hasilnya akan terlihat seperti pada gambar 2 berikut :



Gambar 2: Akses ke *Web Service* Sudah Dapat Dilakukan

4. Selesai.

3 PROSEDUR OPERASI

Sebagaimana terlihat pada gambar 2 bahwa ada 3 (tiga) layanan *web service* yang diberikan yang dapat dilakukan dengan perintah melalui URL *web* seperti berikut :

1. *Inquiry*
2. Pencatatan Pembayaran
3. *Reversal* Pembayaran

Dengan menggunakan *web service* maka hanya ada 2 (dua) informasi dasar yang terjadi, yaitu berupa *request* dan respon dalam dalam kata lain adalah *input* dan *output*. Penjelasan lebih rinci dari tiap layanan yang diberikan adalah sebagai berikut :

3.1 *Inquiry*

Untuk *inquiry* data tagihan, format *request* yang dikirimkan adalah sebagai berikut :

```
1 /sppt/{nop}/{thn}
```

Dengan keterangan sebagai berikut :

- Parameter {`nop`} digantikan dengan Nomor Objek Pajak (NOP) untuk mengidentifikasi objek mana yang akan diambil informasinya.
- Parameter {`thn`} digantikan dengan Tahun Pajak untuk objek pajak yang akan diakses informasinya.

Hasil dari *inquiry* di atas akan menghasilkan beberapa respon dari *server*. Karena beberapa sebab, *server* mungkin saja tidak dapat melakukan proses terhadap *request* yang datang, sehingga diperlukan informasi yang cukup jelas sebagai informasi kepada *client* bahwa *request* yang dikirimkan berhasil atau gagal.

Respon yang dikirimkan oleh *server* akan menjadi bermacam-macam sesuai kondisinya, untuk memudahkan penjelasan, maka akan dibagi menjadi beberapa skenario berikut :

- Skenario *Inquiry* Yang Sukses
- Skenario *Inquiry* Yang Gagal Karena Tahun Pajak Bukan Angka
- Skenario *Inquiry* Yang Gagal Karena Data Tidak Ditemukan

- Skenario *Inquiry* Yang Gagal Karena Kesalahan Server

Secara rinci akan dijelaskan sebagai berikut :

3.1.1 Skenario *Inquiry* Yang Sukses

Pada saat *client* melakukan *request inquiry* data ke *server*, jika memang data tersebut ada pada basis data dan komunikasi berjalan sebagaimana mestinya, maka *server* akan memberikan respon sebagai berikut :

```
{
  "code": 01,
  "message":"Data ditemukan",
  "sppt": {
    "nop":{nop},
    "thn":{thn},
    "nama":{nama wp},
    "alamatOp":{alamat op},
    "pokok":{pokok},
    "denda":{denda}
  }
}

```

Dengan keterangan sebagai berikut :

- `"code"`, berisi kode status respon dari *server* yang menggambarkan suatu kondisi, jika kondisi *inquiry* selesai di proses tanpa ada masalah, maka akan bernilai `01`.
- `"message"`, berisi pesan informasi dari kondisi respon *server* yang telah terjadi, untuk menjelaskan kode status yang tertera pada bagian `"code"`.
- `"sppt"`, berisi informasi *inquiry* yang di*request* oleh *client*.

- {`nop`}, nantinya akan terisi oleh Nomor Objek Pajak PBB-P2 yang di*request* oleh *client*.
- {`thn`}, nantinya akan terisi oleh tahun pajak untuk nomor objek pajak yang di*request* oleh *client*.
- {`nama wp`}, nantinya akan terisi oleh nama wajib pajak untuk nomor objek pajak dan tahun pajak yang diminta oleh *client*.
- {`alamat op`}, nantinya akan terisi oleh alamat objek pajak untuk nomor objek pajak dan tahun pajak yang diminta oleh *client*.
- {`pokok`}, nantinya akan terisi oleh besarnya pokok pajak terhutang.
- {`denda`}, nantinya akan terisi oleh besarnya denda administrasi yang harus dibayarkan oleh wajib pajak. Bila tanggal jatuh tempo belum terlewati, maka akan bernilai 0 (nol), namun apabila tanggal *request* terjadi setelah tanggal jatuh tempo, maka nilainya akan muncul.

3.1.2 Skenario *Inquiry* Yang Gagal Karena Tahun Pajak Bukan Angka

Bila ternyata karena kesalahan *entry* data, tiba-tiba ada karakter bukan angka terselip pada parameter `tahun` pada saat *request inquiry* data, maka *server* akan menghasilkan respon berikut :

```
1 {
2   "code":36,
3   "message: "Tahun Pajak Mengandung Karakter bukan Angka",
4   "sppt":null
5 }
6
```

Keterangan dari kode di atas adalah sebagai berikut :

- `c̈ode¨`, ini berisi kode pesan respon yang dihasilkan *server* setelah menerima *request* dari *client*. Kode ini akan menyimbolkan sesuatu yang terjadi di *server* untuk kemudian dikembalikan sebagai respon ke *client*.

- `m̈essage¨`, berisi keterangan untuk menjelaskan mengenai kode yang dihasilkan pada bagian `c̈ode¨`. Pada bagian ini dijelaskan bahwa ternyata *client* telah mengirimkan parameter tahun pajak yang mengandung karakter bukan angka. Sebagai contoh, misalkan bahwa *client* ternyata mengirimkan parameter tahun pajak dengan nilai `2p16`, sehingga *server* akan mengirimkan respon seperti diatas.

- `s̈ppt¨`, akan berisi `null` yang boleh diartikan kosong, karena data *inquiry* yang diminta oleh *client* memiliki kesalahan sehingga datanya tidak dapat diproses.

3.1.3 Skenario *Inquiry* Yang Gagal Karena Data Tidak Ditemukan

Pada skenario ini, *client* dengan benar memberikan parameter Nomor Objek Pajak dan tahun pajak yang diinginkan tanpa ada kesalahan pengetikan, namun ternyata setelah data dicari dalam basis data dengan menggunakan parameter yang disediakan, data tersebut tidak ada dalam basis data, sehingga respon yang dihasilkan *server* akan terlihat sebagai berikut :

```
1 {
2   "code":10,
3   "message":"Data Tidak Ditemukan",
4   "sppt":null
5 }
6
```

- `c̈ode¨`, bagian ini akan berisi kode pesan yang dihasilkan *server* sebagai

respon terhadap *request* dari *client*. Kode yang dihasilkan untuk skenario ini adalah `10`.

- `m̈essage¨`, bagian ini akan berisi deskripsi atau penjelasan dari kode yang dihasilkan pada bagian `c̈ode¨`. Untuk skenario ini, pesan yang dihasilkan adalah `D̈ata Tidak Ditemukan¨` karena memang data yang diminta oleh *client* tidak ada pada basis data.

- `s̈ppt¨`, karena proses *inquiry* tidak berhasil dengan data yang tidak ditemukan dalam basis data, maka isi dari parameter `s̈ppt¨` ini pun akan bernilai `null` yang artinya kosong.

3.1.4 Skenario *Inquiry* Yang Gagal Karena Kesalahan Server

Pada skenario kali ini, *client* dengan benar mengirimkan parameter *request* ke *server*, namun kegagal terjadi pada saat *server* aplikasi akan mengambil data pada sistem basis data. Respon yang dihasilkan oleh *server* untuk kondisi seperti ini adalah sebagai berikut :

```
1 {
2   "code":04,
3   "message":"Kesalahan DB",
4   "sppt":null
5 }
6
```

Penjelasan untuk kode di atas adalah sebagai berikut :

- `c̈ode¨`, bagian ini akan berisi status kode respon yang diberikan oleh *server* pada *client*, karena terjadi kesalahan komunikasi antara *server* aplikasi dengan *server* basis data, sehingga kode yang dihasilkan adalah kode `04`.

- ẗmessage¨, bagian ini akan berisi penjelasan dari kode yang diberikan di-atas, informasi yang diberikan yaitu *K̈esalahan DB¨*, yang dimaksud adalah adanya kesalahan yang terjadi antara *server* aplikasi yang melakukan *request* data ke *server* basis data.

- s̈ppt¨, tentunya parameter ini akan bernilai `null` karena tidak ada data PBB-P2 yang bisa dikirimkan ke *client*.

3.2 Pencatatan Pembayaran

Untuk pencatatan pembayaran, format *request* yang dapat dikirimkan oleh *client* ke *server* adalah sebagai berikut :

```
1 /bayar/{nop}/{thn}/{tglBayar}/{jamBayar}
2
```

Keterangan untuk format *request* di atas adalah sebagai berikut :

- Parameter {`nop`}, nantinya diisikan dengan Nomor Objek Pajak yang akan dicatatkan pembayarannya.

- Parameter {`thn`}, nantinya diisikan dengan tahun pajak untuk nomor objek pajak yang akan dicatatkan pembayarannya.

- Parameter {`tglBayar`}, nantinya diisikan dengan tanggal terjadinya transaksi pembayaran.

- Parameter {`jamBayar`}, nantinya diisikan dengan jam terjadinya transaksi pembayaran.

Dari *request* pencatatan transaksi pembayaran yang terjadi seperti di atas, hasil respon yang dikirimkan *server* ke *client* akan terbagai menjadi beberapa skenario berikut :

- Skenario Pencatatan Transaksi Pembayaran Yang Sukses.
- Skenario Pencatatan Transaksi Pembayaran Yang Gagal Karena Tanggal dan Jam Pembayaran Telah Melewati Tanggal dan Jam Pencatatan.
- Skenario Pencatatan Transaksi Pembayaran Yang Gagal Karena Tagihan Telah Terbayar Atau Bernilai Nihil.
- Skenario Pencatatan Transaksi Pembayaran Yang Gagal Karena Tagihan Telah Dibatalkan.
- Skenario Pencatatan Transaksi Pembayaran Yang Gagal Karena Terjadi Kesalahan di Sisi Server.

Secara rinci, setiap skenario akan dijelaskan pada bagian-bagian berikut :

3.2.1 Skenario Pencatatan Transaksi Pembayaran Yang Sukses

Saat *client* mengirimkan *request* pencatatan transaksi pembayaran yang benar dengan data tagihan pada sistem basis data ada dan belum terbayar, maka *server* akan menghasilkan respon sebagai berikut :

```
{
  "code":01,
  "message":"Pembayaran Telah Tercatat",
  "byrSppt":{
    "nop":{nop},
    "thn":{thn},
    "ntpd":{ntpd},
    "mataAnggaranPokok":{mataAnggaranPokok},
    "pokok":{pokok},
    "mataAnggaranSanksi":{mataAnggaranSanksi},
    "sanksi":{sanksi},
    "namaWp":{namaWp},
```

```
13      "alamatOp":{alamatOp}
14    }
15 }
16
```

Penjelasan dari kode respon *respon* di atas adalah sebagai berikut :

- Parameter `c̈ode¨` akan berisi kode status respon dari *server*, dalam kondisi skenario ini akan bernilai `01`.
- Parameter `m̈essage¨` akan berisi penjelasan dari kode status pada parameter `c̈ode¨`. Untuk skenario ini, parameter `m̈essage¨` akan berisi `Pembayaran Telah Tercatat`.
- Parameter `b̈yrSppt¨` yang isinya adalah informasi hasil pencatatan transaksi pembayaran pada basis data.
- Parameter {`nop`}, nantinya akan terisi oleh Nomor Objek Pajak untuk PBB-P2 yang dicatatkan pembayarannya.
- Parameter {`thn`}, nantinya akan terisi oleh tahun pajak untuk nomor pajak PBB-P2 yang dicatatkan pembayarannya.
- Parameter {`ntpd`}, nantinya akan terisi oleh Nomor Transaksi Pajak Daerah yang berupa kode unik untuk tiap transaksi yang terjadi.
- Parameter {`mataAnggaranPokok`}, nantinya akan terisi informasi kode mata anggaran penerimaan untuk pokok pajak PBB-P2.
- Parameter {`pokok`}, nantinya akan terisi besarnya pokok pajak PBB-P2 yang terbayar
- Parameter {`mataAnggaranSanksi`}, nantinya akan terisi informasi kode mata anggaran penerimaan untuk sanksi administratif berupa denda pajak.

- Parameter {`sanksi`}, nantinya akan terisi besarnya sanksi administratif, atau dendan pajak yang terbayar untuk PBB-P2.
- Parameter {`namaWp`}, nantinya akan terisi dengan nama wajib pajak untuk nomor objek pajak pada tahun pajak yang di *request* oleh *client*.
- Parameter {`alamatOp`}, nantinya akan terisi dengan alamat objek pajak.

3.2.2 Skenario Pencatatan Transaksi Pembayaran Yang Gagal Karena Tanggal dan Jam Pembayaran Telah Melewati Tanggal dan Jam Pencatatan

Pada saat *client* mengirimkan *request* pencatatan transaksi pembayaran, maka *client* akan mengirimkan informasi tanggal dan jam terjadinya transaksi pembayaran, namun apabila tanggal dan jam terjadinya pembayaran telah melewati tanggal dan jam pencatatan, maka kondisinya akan gagal dicatatkan dan *server* akan mengirimkan respon berikut :

```
1 {
2   "code":32,
3   "message":"Tanggal atau jam pada saat dibayarkan melebihi tanggal
      dan jam saat ini",
4   "byrSppt":null
5 }
6
```

Penjelasan untuk kode di atas adalah sebagai berikut :

- Parameter "`code`", berisi kode status pencatatan pembayaran yang terjadi di *server*, pada skenario ini akan berisi `32`.
- Parameter "`message`", berisi pesan yang menjelaskan kode status yang berada pada parameter "`code`" di atas, pada skenario ini

akan berisi pesan `Tanggal atau jam pada saat dibayarkan melebihi tanggal dan jam saat ini`.

- Parameter `b̈yrSppt¨`, karena proses pencatatan transaksi pembayaran pada skenario ini tidak berhasil, maka isi dari parameter ini adalah `null`.

3.2.3 Skenario Pencatatan Transaksi Pembayaran Yang Gagal Karena Tagihan Telah Terbayar Atau Bernilai Nihil

Pada saat *client* melakukan *request* pencatatan transaksi pembayaran ke *server*, ternyata parameter tanggal dan jam pembayaran telah melewati tanggal dan jam pencatatan (hari dan jam saat terjadinya *request* ini), maka *server* akan mengirimkan pesan respon sebagai berikut :

```
1 {
2   "code":32,
3   "message":"Tanggal atau jam pada saat dibayarkan melebihi tanggal
     dan jam saat ini",
4   "byrSppt":null
5 }
6
```

Penjelasan untuk pesan respon di atas adalah sebagai berikut :

- Parameter `c̈ode¨` adalah kode status respon yang dikirimkan oleh *server* untuk menggambarkan hasil proses dari *request* yang telah dikirimkan oleh *client*.

- Parameter `m̈essage¨` adalah pesan penjelas yang mendeskripsikan kode yang ada pada parameter `c̈ode¨`. Pada skenario ini, akan muncul pesan berikut : `Tanggal atau jam pada saat dibayarkan melebihi tanggal dan jam saat ini`.

- Parameter "byrSppt" akan bernilai null karena memang *request* tidak berhasil diselesaikan oleh *server* karena ada kesalahan isian parameter.

3.2.4 Skenario Pencatatan Transaksi Pembayaran Yang Gagal Karena Tagihan Telah Dibatalkan

Pada skenario ini, *client* mengirimkan *request* pencatatan transaksi pembayaran ke *server*, yang ternyata tagihan untuk nomor objek pajak dan tahun pajak yang diminta sudah dibatalkan, sehingga tidak ada tagihan pajak terhutang. *Server* kemudian akan mengirimkan respon sebagai berikut :

```
1 {
2   "code":03,
3   "message":"Tagihan SPPT Telah Dibatalkan",
4   "byrSppt":null
5 }
6
```

Penjelasan untuk kode respon di atas adalah sebagai berikut :

H Parameter "code" berisi kode status respon *server* terhadap *request* dari *client*.

H Parameter "message" berisi penjelasan atas kode status yang tertera pada parameter "code".

H Parameter "byrSppt" berisi null karena memang proses pencatatan transaksi pembayaran gagal sehingga tidak ada informasi mengenai objek pajak yang dikirimkan ke *client*.

3.2.5 Skenario Pencatatan Transaksi Pembayaran Yang Gagal Karena Terjadi Kesalahan di Sisi Server

Pada saat *client* mengirimkan *request* pencatatan transaksi pembayaran, format dan isi dari tiap parameter masukkan sudah benar, hanya saja pada saat *server* ap-

likasi akan mengambil data pada *server* basis data mengalami kegagalan. Berikut respon yang dikirimkan oleh *server* sebagai jawaban atau respon dari *request client* :

```
{
  "code":04,
  "message":"Kesalahan Server",
  "byrSppt":null
}

```

Penjelasan untuk isi kode di atas adalah sebagai berikut :

- Parameter "code" berisi kode status respon *server* terhadap *request* dari *client*. Pada skenario ini berisi 04.
- Parameter "message" berisi penjelasan dari kode status respon yang berada pada parameter "code", pada skenario ini berisi "Kesalahan Server", karena memang *server* gagal memproses data yang dikirimkan oleh *client*.
- Parameter "byrSppt" berisi null karena memang proses pencatatan transaksi pembayaran gagal di proses.

3.3 *Reversal* Pembayaran

Reversal Pembayaran dilakukan apabila ada kesalahan pencatatan pembayaran yang sebelumnya terjadi. Untuk melakukan *request reversal* pembayaran ini, format yang dikirimkan adalah sebagai berikut :

```
/reversal/{nop}/{thn}/{ntpd}

```

Penjelasan untuk format *request* di atas adalah sebagai berikut :

- Parameter {`nop`} digantikan dengan nomor objek pajak yang pencatatan pembayarannya akan dilakukan *reversal*.
- Parameter {`thn`} digantikan dengan tahun pajak untuk nomor objek pajak yang pencatatan pembayarannya akan dilakukan *reversal*.
- Parameter {`ntpd`} digantikan dengan nomor transaksi pajak daerah yang didapatkan pada saat melakukan *request* pencatatan transaksi pembayaran.

Respon dari *server* untuk *request* di atas akan terbagi menjadi beberapa skenario sebagai berikut :

- Skenario *Reversal* Pembayaran Yang Sukses
- Skenario *Reversal* Pembayaran Yang Gagal Karena Data Yang Diminta Tidak Ada.
- Skenario *Reversal* Pembayaran Yang Gagal Karena Ada Data Pembayaran Yang Tercatat Ganda.
- Skenario *Reversal* Pembayaran Yang Gagal Karena Kesalahan *Server*

Secara rinci, tiap skenario di atas akan dijelaskan pada bagian-bagian berikut :

3.3.1 Skenario *Reversal* Pembayaran Yang Sukses

Pada skenario ini, *client* mengirimkan *request* atau permintaan *reversal* pembayaran karena terjadi kesalahan pencatatan pembayaran sebelumnya. *Request* yang dikirimkan *client* ke *server* adalah seperti dijelaskan diatas, kemudian *server* memberikan respon yang menyatakan bahwa proses *reversal* telah selesai dilaksanakan dan berhasil, berikut kode respon yang dikirim oleh *server* ke *client* :

```
{
  "code":01,
  "message":"Proses Reversal Berhasil",
  "revPembayaran":{
    "nop":{nop},
    "thn":{thn},
    "ntpd":{ntpd}
  }
}

```

Penjelasan untuk kode respon di atas adalah sebagai berikut :

- Parameter `code`, berisi kode status respon yang dikirimkan dari *server* ke *client* yang menandakan berhasil atau tidaknya proses *reversal* dilakukan di sisi *server*.
- Parameter `message`, berisi penjelasan dari kode status respon pada parameter `code`. Pada skenario ini isinya berupa pesan `Proses Reversal Berhasil`.
- Parameter `revPembayaran`, berisi informasi transaksi mana yang berhasil dilakukan *reversal*.
- Parameter {`nop`}, berisi nomor objek pajak yang akan dilakukan *reversal* pembayaran.
- Parameter {`thn`}, berisi tahun pajak untuk nomor objek pajak yang akan dilakukan *reversal* pembayaran.
- Parameter {`ntpd`}, berisi nomor transaksi pajak daerah untuk objek pajak dan tahun pajak yang akan dilakukan *reversal* pembayaran.

3.3.2 Skenario *Reversal* Pembayaran Yang Gagal Karena Data Yang Diminta Tidak Ada

Pada skenario ini, *client* mengirimkan permintaan *reversal* pembayaran ke *server*, namun setelah *server* aplikasi melakukan pemeriksaan data pada *server* sistem basis data, data yang diminta untuk dilakukan *reversal* pembayaran tidak ada, sehingga *server* akan memberikan respon berikut :

```
1 {
2   "code":10,
3   "message":"Data Yang Diminta Tidak Ada",
4   "revPembayaran":null
5 }
6
```

Penjelasan untuk kode respon di atas adalah sebagai berikut :

- Parameter `code`, berisi kode status respon yang dikirimkan *server* ke *client* yang menandakan berhasil atau tidaknya proses *reversal* dilakukan di sisi *server*.
- Parameter `message`, berisi penjelasan atas kode yang muncul pada parameter `code`, untuk skenario ini berisi pesan `Data Yang Diminta Tidak Ada`.
- Parameter `revPembayaran`, berisi informasi objek pajak, tahun pajak, dan nomor transaksi *reversal* pembayaran yang telah dilakukan, karena pada skenario ini proses *reversal* gagal dilakukan, maka bernilai `null`.

3.3.3 Skenario *Reversal* Pembayaran Yang Gagal Karena Ada Data Pembayaran Yang Tercatat Ganda

Pada skenario ini, *client* mengirimkan permintaan *reversal* pembayaran ke *server*, namun *server* gagal melakukan proses *reversal* karena pada saat pencatatan

transaksi pembayaran terjadi pencatatan ganda dengan nomor objek pajak, tahun pajak, dan nomor transaksi pajak daerah yang sama. *Server* kemudian akan mengirimkan pesan respon sebagai berikut :

```
{
  "code":04,
  "message":"Data tersebut Ganda",
  "revPembayaran":null
}

```

Penjelasan untuk kode respon di atas adalah sebagai berikut :

- Parameter `code`, berisi kode status respon dari *server* yang menandakan status berhasil atau tidaknya proses *reversal* pembayaran dilakukan.
- Parameter `message`, berisi pesan penjelas atas kode yang terisi pada parameter `code`, yang pada skenario ini berisi `Data tersebut Ganda`.
- Parameter `revPembayaran`, yang isinya bernilai `null` karena memang proses *reversal* pembayaran gagal dilakukan.

3.3.4 Skenario *Reversal* Pembayaran Yang Gagal Karena Kesalahan *Server*

Pada skenario ini, *client* akan mengirimkan permintaan *reversal* pembayaran ke *server*, namun saat *server* aplikasi melakukan komunikasi dengan *server* sistem basis data, terjadi gangguan sehingga proses *reversal* pembayaran tidak dapat dilanjutkan. Pada kondisi ini, *server* akan mengirimkan respon sebagai berikut :

```
{
  "code":04,
  "message":"Kesalahan Server",
  "revPembayaran":null
```

```
5 }
6
```

Penjelasan untuk kode respon di atas adalah sebagai berikut :

- Parameter `code`, berisi kode status respon dari *server* yang menandakan status berhasil atau tidaknya proses *reversal* pembayaran dilakukan.
- Parameter `message`, berisi pesan penjelas atas kode yang terisi pada parameter `code`, yang pada skenario ini berisi `Kesalahan Server`.
- Parameter `revPembayaran`, yang isinya bernilai `null` karena memang proses *reversal* pembayaran gagal dilakukan.

4 REFERENSI KODE

Dari tiap respon yang dikirimkan oleh *server* ke *client* sebagai jawaban atas tiap *request*, maka *server* akan mengirimkan kode pada parameter `code`. Isi dari parameter `code` ini dapat dilihat pada daftar berikut beserta penjelasannya :

- `01`, artinya permintaan proses yang dikirimkan oleh *client* ke *server* berhasil dilakukan dengan sempurna.
- `10`, artinya data nomor objek pajak untuk tahun pajak yang diinginkan oleh *client* tidak ada pada sistem basis data.
- `13`, artinya nomor objek pajak untuk tahun pajak yang diminta oleh *client* sudah pernah dilakukan pembayaran sebelumnya.
- `03`, artinya nomor objek pajak untuk tahun pajak yang diminta oleh *client* bernilai nihil, tidak ada nilai pajak terhutang.

- `04`, artinya *server* aplikasi tidak dapat berkomunikasi dengan *server* sistem basis data.
- `05`, artinya *server* tidak berhasil melakukan pembentukan nomor transaksi pajak daerah.
- `31`, artinya parameter jumlah pembayaran yang dikirimkan oleh *client* mengandung karakter bukang angka.
- `32`, artinya parameter tanggal dan jam pembayaran telah melewati tanggal dan jam pencatatan
- `33`, artinya parameter tanggal dan jam terjadinya permintaan *reversal* sudah melebihi 24 (dua puluh empat) jam.
- `34`, artinya kode *request* yang dikirimkan oleh *client* tidak benar dan tidak dapat diproses.
- `35`, artinya masa pajak yang dikirimkan oleh *client* sebagai sebuah *request* memiliki karaketer bukan angka.
- `36`, artinya parameter tahun pajak yang dikirim oleh *client* mengandung karakter bukang angka.