RANCANGAN SISTEM *WEB SERVICES* SEBAGAI CARA KOMUNIKASI DENGAN TEMPAT PEMBAYARAN DALAM PENCATATAN PEMBAYARAN PAJAK BUMI DAN BANGUNAN PERDESAAN DAN PERKOTAAN DI KABUPATEN BREBES.

DD MMM 2016

Priyanto Tamami, S.Kom.

1 TUJUAN SISTEM

Tujuan dari dibangunnya sistem *web services* ini adalah mempermudah pencatatan transaksi pembayaran yang terjadi melalui Bank agar tersimpan pada basis data Sistem Manajemen Informasi Objek Pajak PBB-P2.

2 PEMODELAN SISTEM

Sistem akan dimodelkan sebagai bentuk *web services* yang menerima 3 (tiga) bentuk masukan, yaitu untuk melakukan *inquiry*, pencatatan pembayaran, dan *reversal*.

Karena perangkat pemrograman yang digunakan nantinya mendukung pemrograman berorientasi objek, maka akan lebih mudah apabila pendekatan pemodelan menggunakan *Unified Modeling Language* (UML). Bentuk-bentuk diagram yang akan digunakan adalah sebagai berikut :

1. Diagram *Use-Case*

   Diagram ini akan mengilustrasikan gambaran utuh sebuah sistem yang berinteraksi dengan pengguna.

2. Diagram *Activity*

   Diagram ini akan mengilustrasikan aktifitas dari tiap objek yang saling berinteraksi membentuk sebuah sistem yang menerima masukkan, memprosesnya, dan kemudian menghasilkan sebuah keluaran yang dibutuhkan.

3. Diagram *Class*

   Diagram ini akan mengilustrasikan kelas-kelas pembentuk sistem berdasarkan objek-objek yang teridentifikasi sebelumnya.

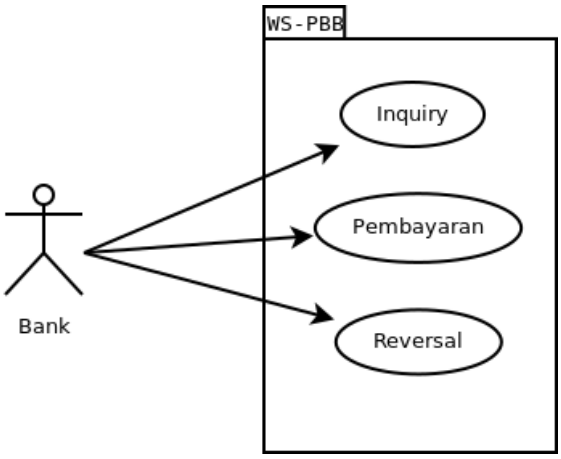
4. Diagram *Sequence*

   Diagram ini akan mengilustrasikan alur interaksi dari tiap kelas berdasarkan skenario tertentu.

Lebih detail mengenai diagram-diagram tersebut akan dijelaskan sebagai berikut.

2.1 Diagram *Use-Case*

Diagram *Use-Case* ini akan menjelaskan gambaran menyeluruh atau gambaran besar aktifitas antara pengguna dengan sistem yang dibangun. Diagram *Use-Case* pada sistem ini seperti terlihat pada gambar 1 :



Gambar 1: Diagram *Use-Case* Sistem *Web Services* Pencatatan Pembayaran PBB-P2

Yang menjadi aktor disini adalah Bank dalam artian *client* yang akan melakukan akses ke sistem *web services* pencatatan pembayaran Pajak Bumi dan Bangunan Perdesaan dan Perkotaan (PBB-P2).

Akses yang dapat dilakukan oleh *client* ada 3 (tiga) skenario, yaitu *Inquiry*, Pembayaran, dan *Reversal*

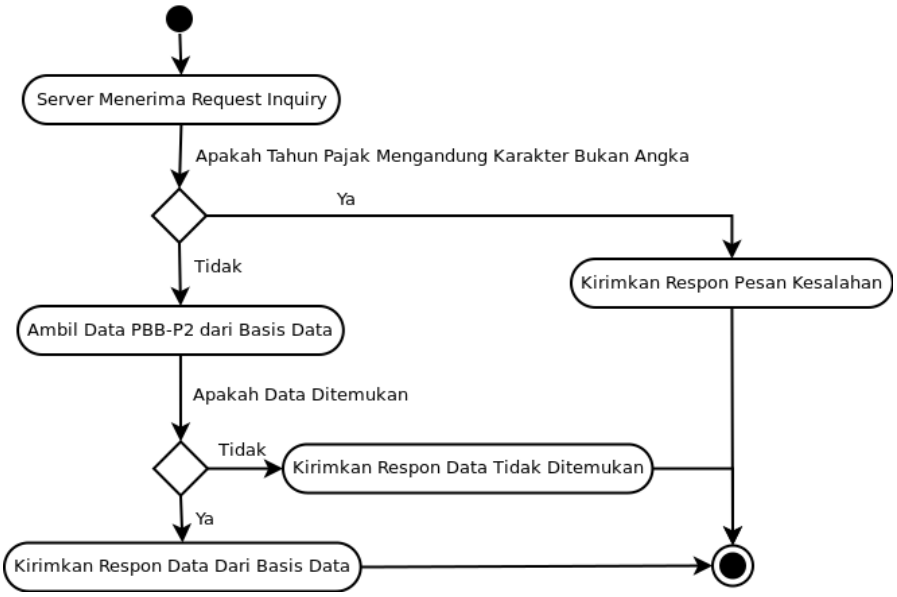
Inquiry ini adalah permintaan informasi PBB-P2 Terhutang oleh *client* ke *server web services*. Skenario Pembayaran adalah permintaan dari *client* ke *server web services* untuk melakukan pencatatan pembayaran PBB-P2 berdasarkan Nomor Objek Pajak (NOP) dan Tahun Pajak yang akan dicatatkan pembayarannya. Sedangkan skenario *reversal* adalah permintaan oleh *client* ke *server web services* untuk melakukan pembatalan pencatatan pembayaran yang telah dilakukan.

2.2 Diagram *Activity*

Diagram *Activity* ini akan menggambarkan alur dari tiap skenario yang terdapat pada diagram *use-case*. Diagramnya akan dijelaskan lebih lanjut pada bagian berikut.

2.2.1 Diagram *Activity* Untuk Skenario *Inquiry*

Diagram *activity* untuk skenario *inquiry* ini akan menjelaskan alur proses dari skenario *inquiry* atau skenario permintaan data oleh *client* berupa informasi tagihan PBB-P2. Diagram *activity* untuk skenario *inquiry* ini seperti terlihat pada gambar 2 :



Gambar 2: Diagram *Activity* Untuk Skenario *Inquiry*

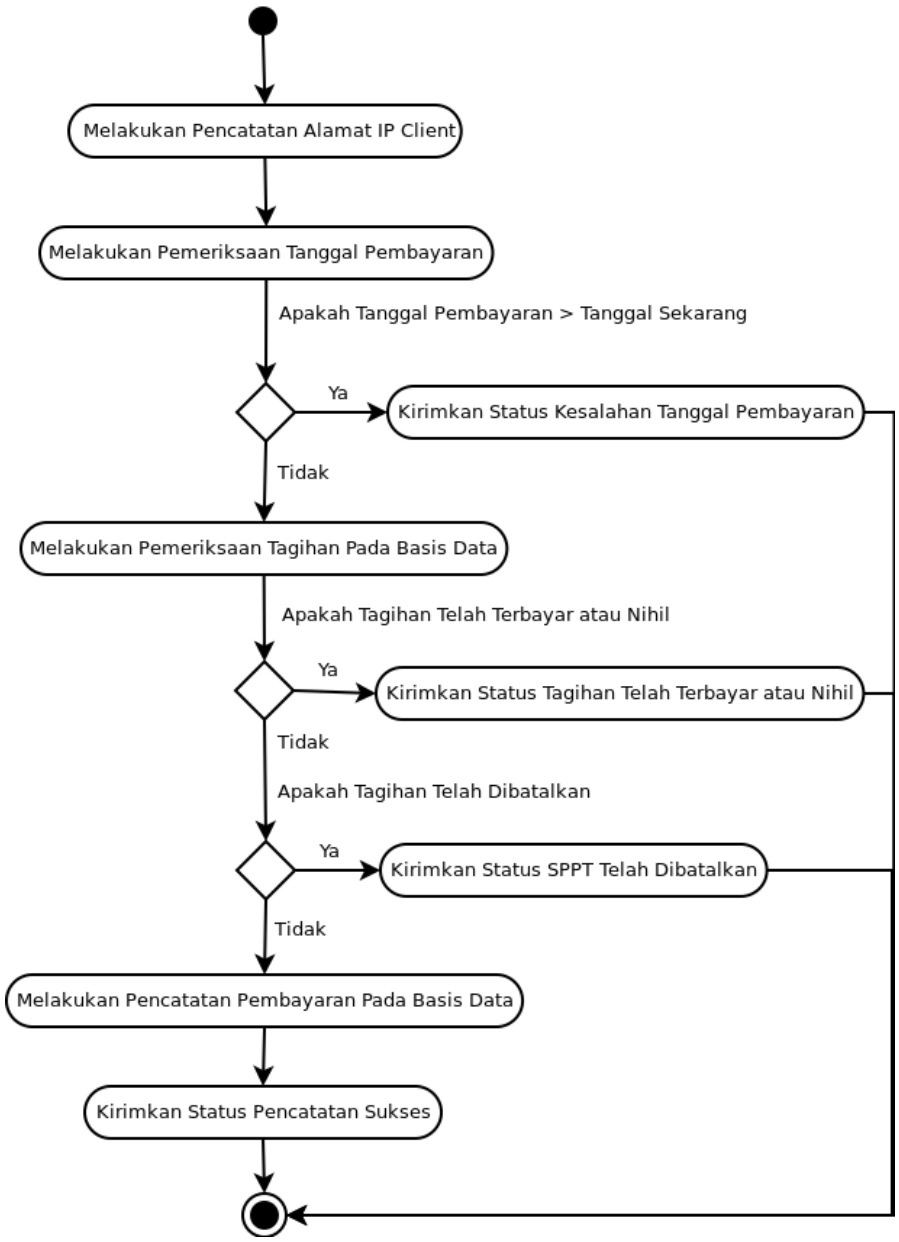
Pada skenario *inquiry* ini, awalnya *server* akan menerima *request inquiry* dari *client*, kemudian *server* melakukan pemeriksaan apakah tahun pajak yang menjadi parameter masukkan permintaan data mengandung karakter selain angka atau tidak, bila mengandung angka, maka *server* akan mengirimkan respon kesalahan ke *client* bahwa tahun pajak yang diminta mengandung karakter bukan angka.

Bila tahun pajak tidak mengandung karaketer bukan angka, maka *server* aplikasi *web services* akan melakukan pengambilan data ke *server* basis data dan dilakukan pengecekan, apakah data yang diminta ada atau tidak, bila tidak ada *server* akan mengirimkan respon kesalahan ke *client* bahwa data yang diminta

tidak ditemukan. Namun bila data ditemukan maka *server* akan mengirimkan respon ke *client* data yang diminta.

2.2.2 Diagram *Activity* Untuk Skenario Pencatatan Pembayaran

Diagram *activity* untuk skenario pencatatan pembayaran akan menjelaskan alur aktifitas pada saat *client* melakukan *request* pencatatan pembayaran ke *server*. Diagram *activity* untuk skenario ini seperti terlihat pada gambar 3 :



Gambar 3: Diagram *Activity* Untuk Skenario Pencatatan Pembayaran

Pertama yang ditangani oleh *server* dari *client* adalah melakukan pencatatan alamat IP *client* untuk keperluan audit.

Kemudian melakukan pemeriksaan terhadap parameter tanggal pembayaran,

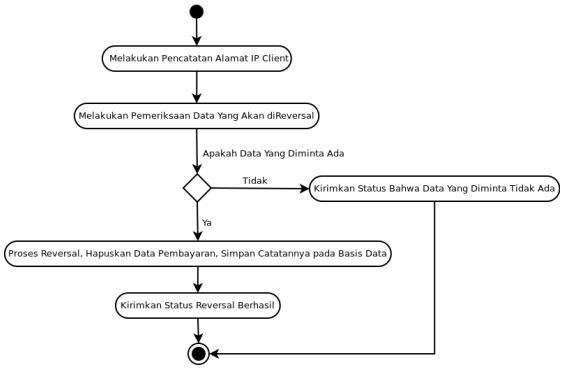
bila tanggal pembayaran lebih dari tanggal saat ini, maka *server* akan mengirimkan respon kesalahan ke *client* karena tidak mungkin tanggal pembayaran terjadi setelah tanggal pencatatan ke SISMIOP.

Selanjutnya adalah melakukan pemeriksaan tagihan pada basis data sesuai parameter yang dikirimkan oleh *client*. Apabila tagihan telah terbayar, atau jumlah tagihan nihil, atau tagihan telah dibatalkan, maka *server* akan mengirimkan respon informasi ke *client* bahwa nomor objek pajak untuk tahun pajak yang diminta telah terbayar, nihil, atau telah dibatalkan.

Langkah setelah melewati serangkaian pemeriksaan parameter tersebut adalah melakukan pencatatan pembayaran pada basis data dan mengirimkan respon informasi ke *client* bahwa pencatatan pembayaran pada basis data telah sukses dan selesai.

2.2.3 Diagram *Activity* Untuk Skenario *Reversal*

Diagram *activity* untuk *reversal* akan menggambarkan rinci dari alur proses *reversal* terjadi. Diagram *activity* untuk skenario *reversal* ini seperti terlihat pada gambar 4 :



Gambar 4: Diagram *Activity* Untuk Skenario *Reversal*

Seperti pada skenario pembayaran, karena transaksi yang dilakukan pada kasus

ini akan merubah isi dari basis data, maka hal pertama yang dilakukan adalah mencatat alamat IP dari *client*.

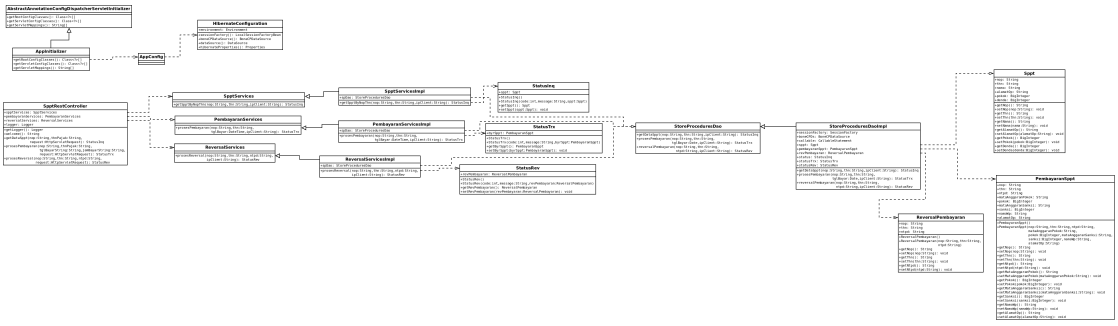
Kemudian melakukan pemeriksaan pada basis data apakah data yang diminta untuk dilakukan *reversal* berdasarkan parameter-parameter yang dikirimkan ada pada basis data atau tidak, bila tidak ada maka *server* akan mengirimkan respon kesalahan bahwa data yang diminta tidak ada.

Bila data yang diminta ada pada basis data, maka proses *reversal* dilakukan, yaitu dengan menghapuskan data pembayaran dan merubah *flag* status pembayaran pada tabel sppt, dan menyimpan catatan penghapusan data pembayaran pada basis data. Terakhir mengirimkan respon ke *client* bahwa proses *reversal* telah selesai dilakukan.

2.3 Diagram *Class*

Pada diagram *class* akan digambarkan hubungan dari tiap kelas hasil implementasi dari objek-objek yang tergambar pada diagram *activity*, namun ada beberapa kelas-kelas yang memang terbentuk karena lingkungan *framework* mengharuskan seperti itu.

Diagram *class* untuk sistem ini seperti terlihat pada gambar 5 :



Gambar 5: Diagram *Class*

Kelas-kelas yang merupakan tuntutan dari *framework* Spring adalah kelas Ap-

pInitializer dan AppConfig. AppInitializer adalah kelas awal yang mewarisi *interface* AbstractAnnotationConfigDispatcherServletInitializer milik Spring yang akan dipanggil pada saat awal aplikasi web akan dijalankan.

Sedangkan kelas AppConfig adalah kelas yang akan menampung konfigurasi aplikasi web nantinya, termasuk disana adalah pemanggilan kelas konfigurasi lainnya, yaitu HibernateConfig yang bertugas melakukan konfigurasi terhadap komunikasi yang terjadi antara aplikasi dengan *server* basis data.

Titik awal dari setiap *request* yang masuk ke *server* akan melalui kelas SpptRestController, dari kelas ini nantinya akan diteruskan ke kelas-kelas *services* yang melayani, seperti kelas SpptServiceImpl, PembayaranServicesImpl, dan ReversalServicesImpl.

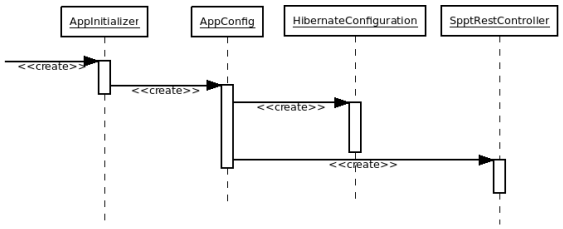
Kelas-kelas *services* ini akan bermuara pada satu kelas *Data Access Object* (DAO) yaitu kelas StoreProceduresDaoImpl. Kelas inilah yang nantinya melakukan akses ke *server* basis data, dan mengembalikannya dalam bentuk objek-objek atau kelas-kelas berikut : Kelas Sppt, untuk menampung informasi tagihan yang tercatat dalam basis data; kelas PembayaranSppt, untuk menampung informasi pencatatan pembayaran yang telah dilakukan dan sukses; yang terakhir adalah kelas ReversalPembayaran, yaitu untuk menampung transaksi *reversal* yang telah sukses.

2.4 Diagram *Sequence*

Diagram *sequence* ini akan menggambarkan interaksi antar kelas untuk tiap alur skenario yang terjadi. Hasil skenario yang terdata adalah sebagai berikut :

2.4.1 Diagram *Sequence* Untuk Skenario Konfigurasi *Spring Framework*

Diagram *sequence* untuk skenario konfigurasi *Spring Framework* akan menjelaskan alur yang terjadi saat *container* melakukan inisialisasi sistem untuk siap menerima *request* dari *client*. Diagram ini seperti terlihat pada gambar **??** :

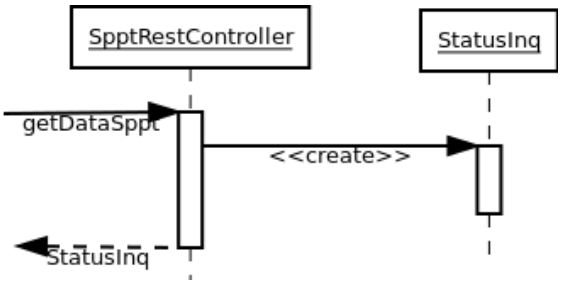


Gambar 6: Diagram *Sequence* Untuk Skenario Konfigurasi *Spring Framework*

Pada saat sistem sudah di *deploy* pada *container*, maka kelas awal yang diperiksa dan dieksekusi adalah kelas AppInitializer, kemudian kelas AppInitializer ini akan melakukan pembentukan instan kelas AppConfig, kemudian kelas AppConfig akan melakukan eksekusi terhadap kelas HibernateConfiguration sebagai penghubung dengan sistem basis data, dan kelas SpptRestController yang melakukan tugasnya sebagai pengontrol tiap *request* yang diterima, dan diteruskan ke bagian-bagian yang menangani.

2.4.2 Diagram *Sequence* Untuk Skenario *Inquiry* Gagal Karena Tahun Pajak Bukan Angka

Diagram *sequence* untuk skenario ini akan menjelaskan alur eksekusi dari awal *request* diterima, sampai kepada kondisi dimana *request* gagal untuk diproses karena parameter tahun pajak memiliki karakter bukan angka. Diagram untuk skenario ini seperti terlihat pada gambar 7 :



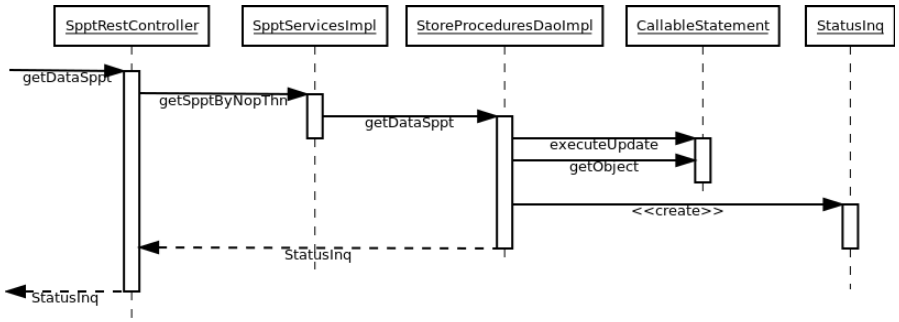
Gambar 7: Diagram *Sequence* Untuk Skenario *Inquiry* Yang Gagal Karena Tahun Pajak Mengandung Karakter Bukan Angka

Setiap *request inquiry* yang masuk ke *server* akan melalui *method* getDataSppt milik kelas SpptRestController, kemudian pada *method* getDataSppt ini akan dilakukan pemeriksaan terhadap parameter tahun pajak yang diinginkan oleh *client*.

Saat ditemukan bahwa parameter tahun pajak mengandung karakter bukan angka, maka akan dibuat instan dari kelas StatusInq sebagai bahan respon terhadap *client* bahwa *request* yang diminta tidak dapat diproses.

2.4.3 Diagram *Sequence* Untuk Skenario *Inquiry* Gagal Karena Data Tidak Ditemukan

Diagram *sequence* ini akan menggambarkan alur komunikasi *inquiry* yang gagal karena data Nomor objek Pajak (NOP) untuk tahun pajak yang diminta tidak ada pada basis data. Diagramnya terlihat seperti pada gambar 8 :



Gambar 8: Diagram *Sequence* Untuk Skenario *Inquiry* Yang Gagal Karena Data Tidak Ditemukan Dalam Basis Data

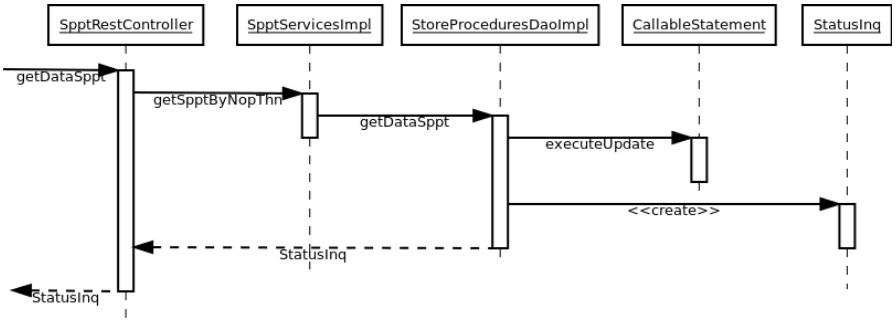
Request yang masuk ke *server* terhadap *inquiry* akan datang ke *method* getDataSppt milik kelas SpptRestController, berdasarkan parameter Nomor Objek Pajak (NOP) dan tahun pajak, *method* ini akan memanggil *method* getSpptByNopThn milik kelas SpptServicesImpl, dari *method* getSpptByNopThn milik kelas SpptServicesImpl akan memanggil *method* getDataSppt milik kelas StoreProceduresDaoImpl.

Selanjutnya di dalam *method* getDataSppt milik kelas StoreProceduresDaoImpl melakukan eksekusi *store procedure* dengan memanggil *method* executeUpdate milik kelas CallableStatement, kemudian mengambil hasil dari pemanggilan *store procedure* pada basis data dengan *method* getObject.

Namun pada saat pemanggilan *method* getObject, data yang diminta tidak ada dalam basis data, sehingga kelas StoreProceduresDaoImpl akan membuat instan dari kelas StatusInq yang berisi informasi bahwa objek pajak untuk tahun pajak yang diinginkan tidak ada pada basis data. Kemudian instan kelas StatusInq ini dikembalikan ke kelas SpptRestController sebagai bahan respon ke *client*.

2.4.4 Diagram *Sequence* Untuk Skenario *Inquiry* Gagal Karena Kesalahan Server

Diagram ini akan menjelaskan alur komunikasi data antar kelas bila ada kesalahan proses saat melakukan pemanggilan *store procedure* pada *server* basis data. Diagram *sequence* untuk skenario ini seperti terlihat pada gambar 9 :



Gambar 9: Diagram *Sequence* Untuk Skenario *Inquiry* Yang Gagal Karena Kesalahan Komunikasi Antara *Server* Aplikasi Dengan *Server* Basis Data

Request yang masuk untuk skenario *inquiry* akan diterima dan diproses oleh *method* getDataSppt milik kelas SpptRestController. Dari *method* ini, kemudian akan dipanggil *method* getSpptByNopThn milik kelas SpptServicesImpl.

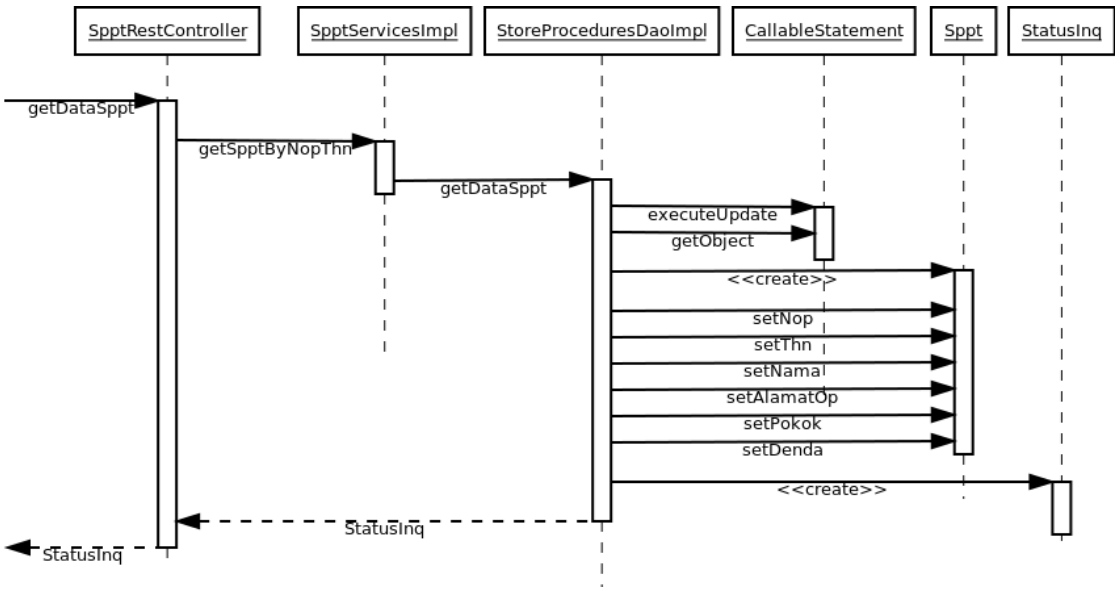
Dari *method* getSpptByNopThn milik kelas SpptServicesImpl, kemudian memanggil *method* getDataSppt milik kelas StoreProceduresDaoImpl, yang kemudian mencoba melakukan eksekusi *store procedure* milik basis data dengan memanggil *method* executeUpdate milik kelas CallableStatement, tetapi eksekusi ini karena beberapa hal yang tidak diketahui, hubungan komunikasi antara *server* Aplikasi dengan *server* basis data terputus, sehingga keluar *exception* dan proses pengambilan atau *inquiry* data tidak dapat diteruskan.

Sampai sini, *method* getDataSppt milik kelas StoreProceduresDaoImpl akan membuat instan dari kelas StatusInq yang berisi informasi kegagalan komunikasi ke *server* basis data, hasil instan dari kelas StatusInq ini dikirimkan kembali ke

kelas SpptRestController yang nantinya memberikan respon ke *client* bahwa proses *inquiry* gagal dilakukan.

2.4.5 Diagram *Sequence* Untuk Skenario *Inquiry* Yang Sukses

Diagram ini akan menjelaskan alur yang terjadi saat *request inquiry* berhasil di respon sesuai data yang ada pada basis data. Diagram *sequence* untuk skenario ini seperti terlihat pada gambar 10 :



Gambar 10: Diagram *Sequence* Untuk Skenario *Inquiry* Yang Sukses

Proses *request inquiry* dari *client* akan berawal dari *method* getDataSppt milik kelas SpptRestController, kemudian memanggil *method* getSpptByNopThn milik kelas SpptServicesImpl, dari *method* ini kemudian memanggil *method* getDataSppt milik kelas StoreProceduresDaoImpl yang kemudian melakukan eksekusi terhadap *store procedures* milik basis data dengan memanggil *method* executeUpdate milik kelas CallableStatement.

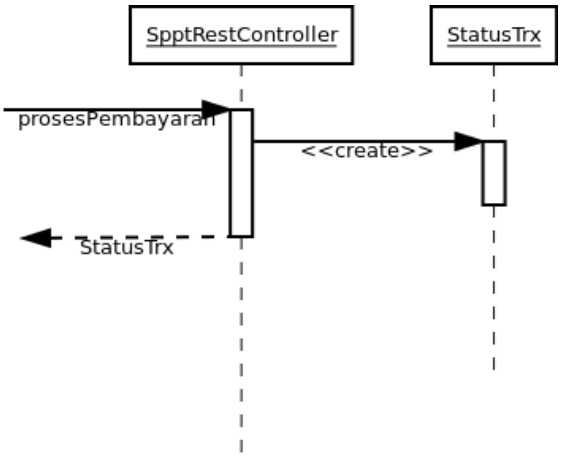
Untuk mendapatkan hasil dari pemanggilan *store procedures* milik basis data,

maka perlu dilakukan pemanggilan *method* getObject milik kelas CallableStatement, yang hasilnya dimasukkan sebagai parameter pembentuk kelas Sppt.

Langkah terakhir adalah membungkus instan kelas Sppt yang telah terisi data dari basis data dengan kelas StatusInq, kemudian instan dari kelas ini dikembalikan ke kelas SpptRestController yang akhirnya dikirimkan ke *client* sebagai respon *inquiry*.

2.4.6 Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Gagal Karena Jam Pembayaran Melebihi Jam Pencatatan

Diagram ini akan menjelaskan alur komunikasi antar kelas pada saat ada *request* pencatatan transaksi pembayaran tetapi gagal karena jam pada saat pembayaran melebihi jam pada saat pencatatan di basis data, karena tidak mungkin transaksi yang terjadi tercatat lebih dulu sebelum terjadinya pembayaran. Diagram *sequence* untuk skenario ini seperti terlihat pada gambar 11 :



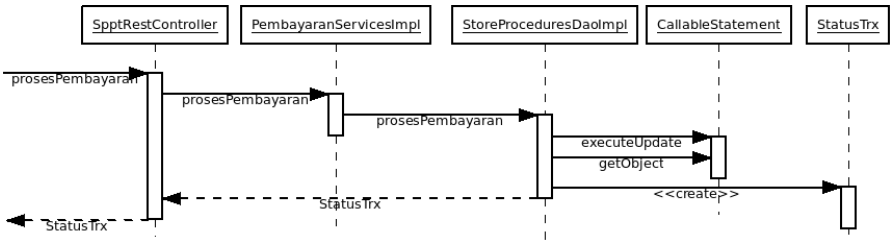
Gambar 11: Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Yang Gagal Karena Jam Pembayaran Melebihi Jam Pencatatan

Request yang datang dari *client* akan langsung memanggil *method* prosesPembayaran milik kelas SpptRestController. Di dalam *method* ini, terdapat pemerik-

saan parameter tanggal pembayaran yang apabila parameter tanggal pembayaran berisi tanggal yang melebihi tanggal saat ini, maka *method* prosesPembayaran akan membentuk instan dari kelas StatusTrx, kemudian mengirimkan informasi kelas StatusTrx ini kepada *client* bahwa tanggal dan jam pembayaran melebihi tanggal dan jam hari saat ini, sehingga proses pencatatan pembayaran tidak dapat dilakukan.

2.4.7 Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Gagal Karena Tagihan Telah Terbayar Atau Nihil

Diagram ini akan menceritakan alur komunikasi antar kelas untuk skenario transaksi pembayaran yang gagal karena tagihan telah terbayar atau memang tagihan untuk nomor objek pajak (NOP) dan tahun pajak tersebut nihil. Diagram *sequence* untuk skenario ini seperti terlihat pada gambar 12 :



Gambar 12: Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Yang Gagak Karena Tagihan Telah Terbayar Atau Nihil

Request client akan langsung diarahkan ke *method* prosesPembayaran milik kelas SpptRestController, kemudian *method* ini akan memanggil *method* prosesPembayaran milik kelas PembayaranServicesImpl, lalu *method* ini akan memanggil *method* prosesPembayaran milik kelas StoreProceduresDaoImpl.

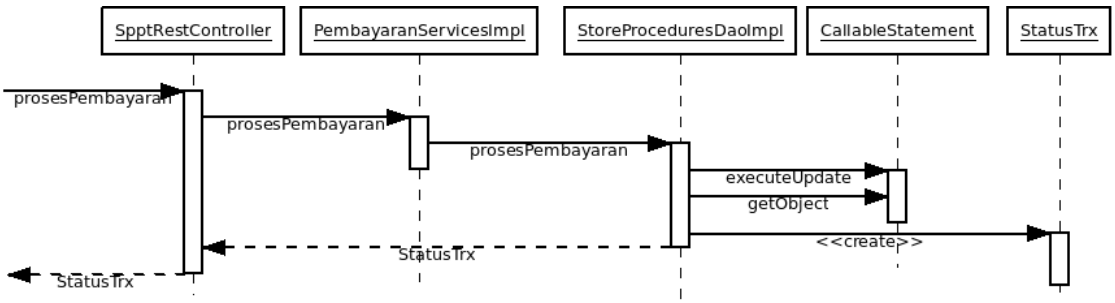
Didalam *method* prosesPembayaran milik kelas StoreProceduresDaoImpl, akan melakukan eksekusi terhadap *store procedures* di sistem basis data dengan

melakukan pemanggilan *method* executeUpdate milik kelas CallableStatement, kemudian untuk mengambil hasil dari eksekusi *store procedure*, dilakukan pemanggilan *method* getObject milik kelas CallableStatement.

Karena datanya telah terbayar, atau tagihannya nihil, maka *store procedure* akan memberikan keterangan kondisi ini yang hasilnya disimpan pada kelas StatusTrx dan dikembalikan ke kelas SpptRestController sebagai bahan respon ke *client*.

2.4.8 Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Gagal Karena Telah Dibatalkan

Desain ini akan menjelaskan gambaran alur komunikasi antar kelas untuk skenario pencatatan transaksi pembayaran yang gagal karena tagihan untuk nomor objek pajak (NOP) dan tahun pajak yang diminta telah dibatalkan. Diagram *Sequence* untuk skenario ini seperti pada gambar 13 :



Gambar 13: Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Yang Gagal Karena SPPT Telah Dibatalkan

Proses *request* pencatatan pembayaran yang datang dari *client* akan langsung masuk ke *method* prosesPembayaran milik kelas SpptRestController, dari *method* ini kemudian akan memanggil *method* prosesPembayaran milik kelas PembayaranServicesImpl.

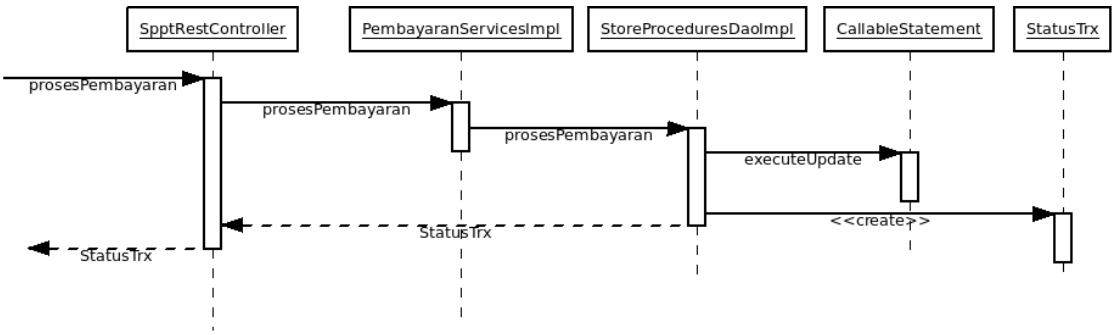
Dalam *method* prosesPembayaran pada kelas PembayaranServicesImpl, akan memanggil *method* prosesPembayaran milik kelas StoreProceduresDaoImpl, dari *method* ini kemudian akan melakukan eksekusi *store procedure* milik basis data dengan memanggil *method* executeUpdate milik kelas CallableStatement, untuk menampung hasil kembalian dari *store procedure* maka digunakan *method* getObject milik CallableStatement.

Hasil kembalian dari pemanggilan *store procedure* memberikan informasi bahwa tagihan untuk nomor objek pajak (NOP) dan tahun pajak yang diminta telah dibatalkan, sehingga tidak ada pajak terhutang atas NOP dan tahun pajak tersebut.

Hasil dari kembalian *store procedure* ini akan ditempatkan pada kelas StatusTrx yang dikembalikan ke kelas SpptRestController sebagai bahan respon ke *client*

2.4.9 Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Gagal Karena Kesalahan Server

Diagram ini akan menjelaskan alur komunikasi untuk menangani skenario pencatatan transaksi pembayaran yang gagal karena ada kesalahan dengan komunikasi antara *server* aplikasi dengan *server* basis data. Diagram *sequence* untuk menggambarkan skenario ini adalah seperti pada gambar 14 :



Gambar 14: Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Yang Gagal Karena Ada Kesalahan Di *Server*

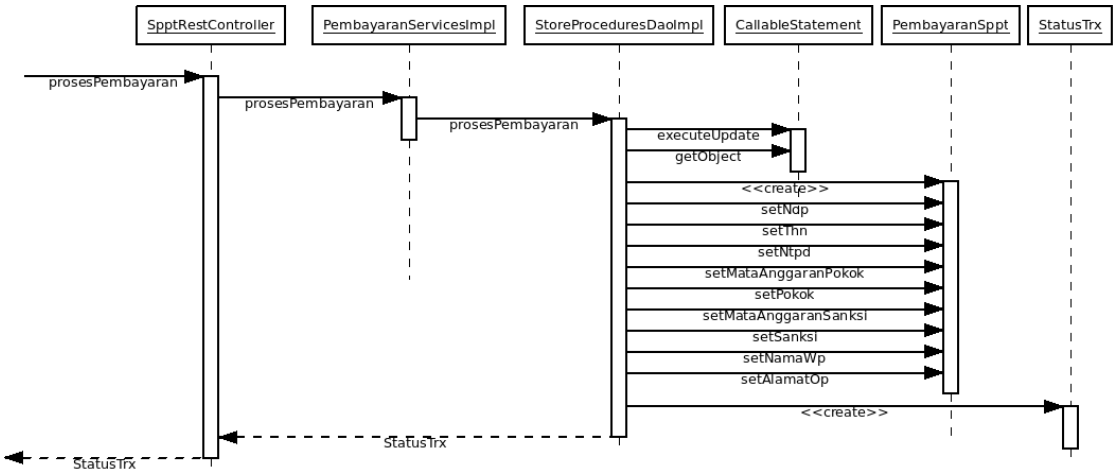
Request pencatatan transaksi pembayaran dari *client* akan langsung ditangani oleh *method* prosesPembayaran milik kelas SpptRestController. *Method* ini kemudian akan melakukan pemanggilan ke *method* prosesPembayaran milik kelas PembayaranServicesImpl, yang didalam *method* ini pun akan memanggil *method* prosesPembayaran dari kelas StoreProceduresDaoImpl.

Dari *method* prosesPembayaran milik kelas StoreProceduresDaoImpl akan diek-sekusi *store procedure* milik basis data dengan memanggil *method* executeUpdate milik kelas CallableStatement, namun karena ada beberapa kendala yang terjadi seperti putusnya koneksi jaringan, atau basis data dalam kondisi *offline*, maka komunikasi antara *server* aplikasi dengan *server* basis data terganggu, sehingga munculah kesalahan disana.

Kesalahan muncul kemudian dicatatkan dalam instan kelas StatusTrx yang kemudian dikembalikan ke kelas SpptRestController sebagai bahan respon ke *client*.

2.4.10 Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Yang Sukses

Diagram ini akan menggambarkan alur komunikasi antar kelas untuk menyelesaikan skenario pencatatan transaksi pembayaran yang sukses. Diagram *sequence* untuk skenario ini seperti terlihat pada gambar 15 :



Gambar 15: Diagram *Sequence* Untuk Skenario Transaksi Pembayaran Yang Sukses

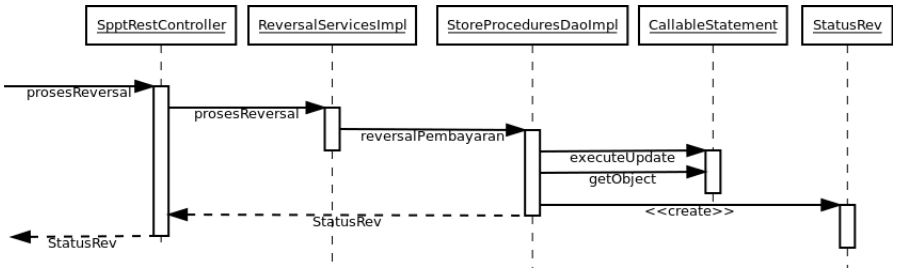
Request dari *client* akan langsung ditangani oleh *method* prosesPembayaran milik kelas SpptRestController, *method* ini kemudian melakukan pemanggilan *method* prosesPembayaran milik kelas PembayaranServicesImpl, selanjutnya *method* prosesPembayaran milik kelas PembayaranServicesImpl akan memanggil *method* prosesPembayaran milik kelas StoreProceduresDaoImpl.

Dalam *method* prosesPembayaran milik kelas StoreProceduresDaoImpl, ada beberapa kegiatan yang dilakukan, yaitu melakukan eksekusi *store procedure* milik basis data dengan memanggil *method* executeUpdate milik kelas CallableStatement, kemudian mengambil hasil dari pemanggilan *store procedure* dengan *method* getObject milik kelas CallableStatement pula, hasilnya disimpan dalam kelas PembayaranSppt.

Kemudian langkah selanjutnya adalah membuat instan dari kelas StatusTrx dan memasukan instan kelas PembayaranSppt didalamnya sebagai informasi ke *client* bahwa proses pencatatan pembayaran telah dilakukan dengan sukses. Hasil instan kelas StatusTrx ini kemudian dikembalikan ke kelas SpptRestController sebagai bahan respon ke *client*.

2.4.11 Diagram *Sequence* Untuk Skenario *Reversal* Gagal Karena Data Yang Diminta Tidak Ada

Diagram ini akan memberikan gambaran bagaimana arus komunikasi terjadi sehingga skenario proses reversal yang gagal karena data yang diminta tidak ada pada sistem basis data. Diagram *sequence* untuk skenario ini seperti terlihat pada gambar 16 :



Gambar 16: Diagram *Sequence* Untuk Skenario *Reversal* Yang Gagal Karena Data Yang Diminta Tidak Ada

Request dari *client* akan diterima oleh *method* prosesReversal milik kelas SpptRestController, kemudian di dalam *method* ini akan memanggil *method* prosesReversal milik kelas ReversalServicesImpl, di dalam *method* prosesReversal milik kelas ReversalServicesImpl pun akan memanggil *method* lain bernama reversalPembayaran milik kelas StoreProceduresDaoImpl.

Di dalam *method* reversalPembayaran milik kelas StoreProceduresDaoImpl akan melakukan eksekusi terhadap *store procedure* melalui pemanggilan *method* executeUpdate, dan menerima responnya dengan menggunakan *method* getObject milik kelas CallableStatement.

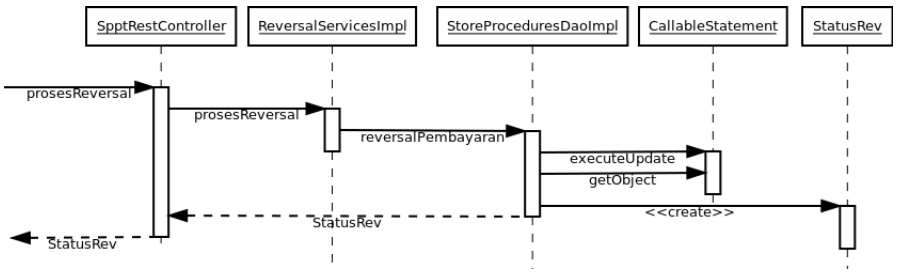
Pada skenario ini, pemanggilan *store procedure* milik sistem basis data akan menggembalikan informasi bahwa data *reversal* yang diminta berdasarkan Nomor Objek Pajak (NOP), Tahun Pajak, dan Nomor Transaksi Pajak Daerah (NTPD) tidak ada pada basis data, sehingga *method* reversalPembayaran milik kelas Store-

ProceduresDaoImpl akan membuat instan dari kelas StatusRev dan mengisikan dengan informasi tersebut.

Instan dari kelas StatusRev kemudian dikembalikan oleh *method* reversalPembayaran milik kelas StoreProceduresDaoImpl ke kelas SpptRestController sebagai bahan respon ke *client*.

2.4.12 Diagram *Sequence* Untuk Skenario *Reversal* Gagal Karena Ada Data Pembayaran Yang Tercatat Ganda

Diagram ini akan menggambarkan skenario proses *reversal* yang gagal karena dalam basis data terdapat pencatatan transaksi pembayaran ganda yang terjadi karena kesalahan sistem. Diagramnya terlihat sama dengan skenario sebelumnya, yaitu data tidak ada pada basis data saat akan dilakukan *reversal*. Diagram untuk skenario ini terlihat seperti pada gambar 17 :



Gambar 17: Diagram *Sequence* Untuk Skenario *Reversal* Yang Gagal Karena Ada Data Pembayaran Yang Tercatat Ganda

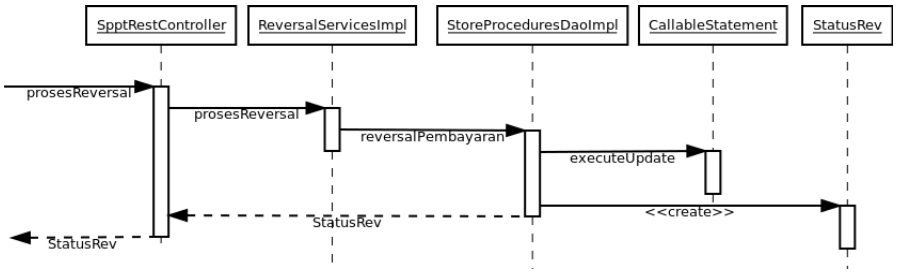
Perbedaan yang terjadi antara skenario ini dengan skenario sebelumnya, yaitu data pembayaran tidak ada pada basis data adalah pada saat pemanggilan *method* getObject milik kelas CallableStatement, hasil dari eksekusi *store procedure* milik sistem basis data akan mengirimkan informasi bahwa data transaksi pembayaran yang tercatat ada lebih dari 1 (satu) data.

Dengan berdasarkan informasi tersebut, kemudian *method* reversalPembayaran

akan membuat instan kelas StatusRev dengan mengisikan informasi seperti hasil yang didapat dari eksekusi *store procedure*, setelah itu mengembalikan instan kelas StatusRev ke kelas SpptRestController sebagai bahan respon ke *client*.

2.4.13 Diagram *Sequence* Untuk Skenario *Reversal* Gagal Karena Kesalahan Server

Diagram ini akan menjelaskan alur komunikasi yang terjadi untuk menjelaskan skenario *reversal* yang gagal karena adanya kesalahan di sisi *server*. Diagram *sequence* untuk menjelaskan hal ini bisa dilihat pada gambar 18 :



Gambar 18: Diagram *Sequence* Untuk Skenario *Reversal* Yang Gagal Karena Terjadi Kesalahan Pada *Server*

Proses diawali dengan masuknya *request client* ke *method* prosesReversal milik kelas SpptRestController. Kemudian *method* ini akan memanggil *method* prosesReversal milik kelas ReversalServicesImpl. Di dalam *method* prosesReversal milik kelas ReversalServicesImpl akan memanggil *method* reversalPembayaran milik kelas StoreProceduresDaoImpl.

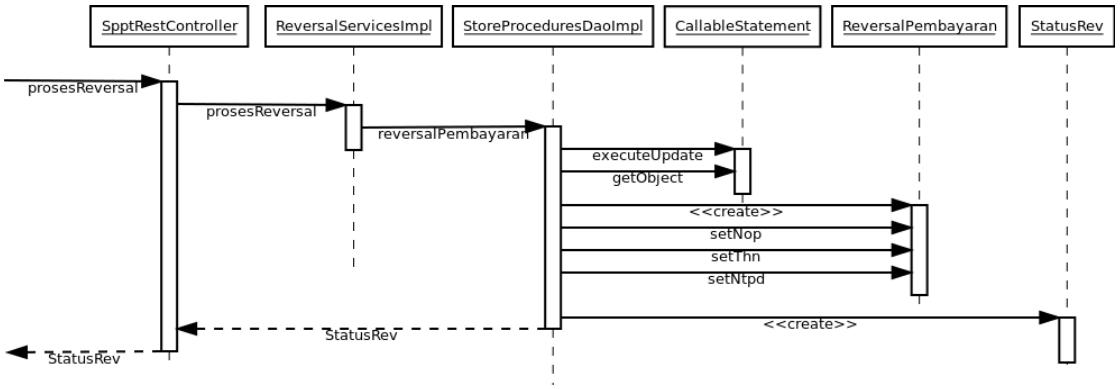
Pada *method* reversalPembayaran milik kelas StoreProceduresDaoImpl, dilakukan eksekusi terhadap *store procedure* milik sistem basis data dengan memanggil *method* executeUpdate milik kelas CallableStatement, tetapi keluar kesalahan (*exception*) sebelum eksekusi berhasil dilakukan. Kemudian *method* reversalPembayaran milik kelas StoreProceduresDaoImpl membuat instan dari kelas Status-

Rev dan mengisinya dengan informasi kesalahan karena gagalnya *server* aplikasi melakukan komunikasi dengan *server* basis data.

Hal terakhir adalah mengembalikan instan dari kelas StatusRev ke kelas SpptRestController sebagai bahan respon ke *client*.

2.4.14 Diagram *Sequence* Untuk Skenario *Reversal* Yang Sukses

Diagram ini akan menjelaskan alur komunikasi antar kelas agar skenario *reversal* yang terjadi sukses dilakukan. Diagram *sequence* untuk skenario ini seperti terlihat pada gambar 19 :



Gambar 19: Diagram *Sequence* Untuk Skenario *Reversal* Yang Sukses

Request untuk proses *reversal* akan memanggil *method* prosesReversal milik kelas SpptRestController. *Method* ini kemudian memanggil *method* prosesReversal milik kelas ReversalServicesImpl.

Di dalam *method* prosesReversal milik kelas ReversalServicesImpl akan memanggil *method* reversalPembayaran milik kelas StoreProceduresDaoImpl. Kemudian di dalam *method* reversalPembayaran ini akan melakukan eksekusi terhadap *store procedure* yang berada pada sistem basis data dengan menggunakan *method* executeUpdate milik kelas CallableStatement.

Respon yang dihasilkan dari eksekusi *store procedure* milik basis data diambil dengan *method* getObject milik kelas CallableStatement. Kemudian hasilnya disimpan dalam instan kelas ReversalPembayaran, setelah itu instan kelas ReversalPembayaran dimasukkan dalam instan kelas StatusRev yang kemudian dikirimkan ke kelas SpptRestController sebagai bahan respon ke *client*.

3 AKTIVITAS PEMROSESAN DATA

Aktivitas pemrosesan data akan terbagi menjadi 3 (tiga) proses inti seperti pada pembahasan diagram *use-case*, yaitu aktivitas *inquiry* data, aktivitas pencatatan pembayaran, dan aktivitas *reversal* pembayaran.

3.1 Aktivitas *Inquiry*

Output terpenting dari aktivitas *inquiry* ini adalah berupa informasi PBB-P2 sebagaimana yang tersimpan pada sistem basis data yang nantinya terdiri dari Nomor Objek Pajak (NOP), Tahun Pajak, Nama Wajib Pajak, Alamat objek Pajak, Besaran Pokok PBB-P2, dan Denda Administrasi bila ada.

Sedangkan data yang diperlukan sebagai parameter *input* adalah Nomor Objek Pajak (NOP) dan Tahun Pajak. Nantinya parameter-parameter ini akan diperiksa, seperti apakah tahun pajak mengandung karakter bukan angka, lalu benarkah NOP yang diminta ada pada basis data, dan beberapa pemeriksaan lain sehingga *output* dapat dihasilkan dari 2 (dua) parameter *input* ini.

3.2 Aktivitas Pencatatan Pembayaran

Output terpenting dari aktivitas ini pada sistem basis data adalah sebagai berikut :

1. Tercatatnya transaksi pembayaran pada tabel PEMBAYARAN_SPPT.

2. Berubahnya isi kolom STATUS_PEMBAYARAN_SPPT pada tabel SPPT, dari karakter 0 (nol) menjadi 1 (satu), yang artinya bahwa NOP pada tahun pajak tersebut telah terbayar.

3. Tercatatnya aktivitas transaksi pembayaran ini pada tabel LOG_TRX_PEMBAYARAN.

Dan *output* sebagai respon dari *request client* adalah sebagai berikut :

1. NOP. Nomor Objek Pajak yang telah dicatatkan pembayarannya.

2. THN. Tahun pajak untuk nomor objek pajak yang dicatatkan pembayarannya.

3. NTPD. Nomor Transaksi Pajak Daerah yang digunakan sebagai nomor pengesahan bahwa sebuah transaksi sudah tercatat pada basis data.

4. Mata Anggaran Pokok, adalah mata anggaran sebagai identitas pencatatan realisasi penerimaan pokok PBB-P2 pada pembukuan keuangan Pemerintah Kabupaten Brebes.

5. Pokok, adalah besarnya pokok PBB-P2 terhutang yang dibayarkan.

6. Mata Anggaran Sanksi, adalah mata anggaran sebagai identitas pencatatan sanksi administrasi atau denda administrasi yang muncul karena pembayaran dilakukan melewati tanggal jatuh tempo.

7. Sanksi, adalah besarnya sanksi administrasi atau denda administrasi yang dibayarkan.

8. Nama Wajib Pajak.

9. Alamat Objek Pajak.

Parameter *input* yang diperlukan untuk dilakukannya aktivitas ini yaitu :

1. NOP, Nomor objek Pajak yang akan dilakukan pencatatan pembayaran.
2. THN, Tahun Pajak untuk NOP yang akan dilakukan pencatatan pembayarannya.
3. Tanggal Pembayaran, tanggal terjadinya pembayaran
4. Jam Bayar, jam terjadinya pembayaran.

Proses yang terjadi tentu saja parameter-parameter *input* yang ada akan dilakukan pemeriksaan dasar terlebih dahulu, setelah itu baru prosesnya akan melakukan eksekusi terhadap *store procedure* yang ada pada sistem basis data dan menghasilkan *output* seperti yang diinginkan diatas.

3.3 Aktivitas *Reversal* Pembayaran

Output yang diharapkan terjadi pada basis data dari aktivitas *reversal* ini yaitu :

1. Terhapusnya pencatatan pembayaran pada tabel PEMBAYARAN_SPPT untuk NOP dan tahun pajak yang diminta.
2. Berubahnya isi kolom STATUS_PEMBAYARAN_SPPT pada tabel SPPT dari 1 (satu) yang artinya sudah bayar ke 0 (nol) menjadi belum bayar, tentunya sesuai dengan NOP dan tahun pajak yang diinginkan.
3. Melakukan pencatatan aktivitas *reversal* pada tabel LOG_REVERSAL.

Sedangkan *output* yang diharapkan untuk merespon *request client* adalah data-data berikut :

1. NOP, Nomor Objek Pajak yang dilakukan *reversal* pembayarannya.

2. THN, tahun pajak untuk nomor objek pajak yang dilakukan *reversal* pembayarannya.

3. NTPD, Nomor Transaksi Pajak Daerah yang telah dilakukan *reversal* atau pembatalan pencatatan pembayarannya.

Parameter *input* yang diperlukan untuk melakukan aktivitas *reversal* ini adalah sebagai berikut :

1. Nomor Objek Pajak yang dilakukan *reversal* pembayarannya.

2. Tahun pajak untuk nomor objek pajak yang akan dilakukan *reversal*.

3. Nomor Transaksi Pajak Daerah untuk konfirmasi transaksi mana yang akan dilakukan *reversal*.

Prosesnya pun seperti biasa akan dilakukan pemeriksaan dasar parameter terlebih dahulu, yang kemudian dilakukan pemanggilan *store procedure* pada sistem basis data sehingga *output* yang diharapkan dapat dihasilkan.