ANALISIS SISTEM INFORMASI PEMBAYARAN PBB

25 Februari 2018

Priyanto Tamami, S.Kom.

1 SASARAN DAN BATASAN SISTEM

Sasaran dari sistem ini adalah mampu menampilkan informasi besaran pajak terhutang dan status pelunasan untuk jenis pajak bumi dan bangunan sektor perdesaan dan perkotaan, sehingga masyarakat wajib pajak tidak perlu lagi menduga-duga apakah pembayaran yang dilakukan telah diterima oleh kas Pemerintah Daerah atau belum.

Batasan dari sistem ini tentu saja yang dapat melakukan akses informasi adalah pengguna yang memiliki noomr objek pajak untuk pajak bumi dan bangunan sektor perdesaan dan perkotaan, kemudian, karena kondisi pencatatan pada peladen di BPPKAD Kabupaten Brebes dilakukan H+1, maka pembayaran atau setoran yang dilakukan di hari bersangkutan tidak akan terlihat perubahan status pembayarannya.

2 ARSITEKTUR SISTEM

Seperti membangun sebuah gedung atau bangunan, agar bangunan dapat berdiri dengan kokoh dan memudahkan dalam pemeliharaan, maka diperlukan sebuah arsitektur, sedangkan pada pembuatan sistem aplikasi maka diperlukan sebuah arsitektur sistem aplikasi yang tujuannya pun agar kokoh dalam artian stabil dengan sedikit permasalahan yang muncul, dan memberikan kemudiahan pada saat pemeliharaan atau penambahan fasilitas pada sistem aplikasi yang sudah jadi.

Arsitektur sistem akan terbagi menjadi 3 (tiga) bagian, yaitu :

a. Bagian Basis Data

b. Bagian Logika Aplikasi

c. Bagian Tampilan Aplikasi

3 DESKRIPSI SUB SISTEM

3.1 Bagian Basis Data

Pada bagian ini akan terdapat beberapa tabel dari aplikasi SISMIOP yang digunakan untuk pengelolaan pajak bumi dan bangunan sektor perdesaan dan perkotaan, karena sasarannya adalah menampilkan informasi pembayaran atas sebuah objek, maka tentu saja sistem basis data adalah sistem basis data produksi yang digunakan sebagai pencatatan manajemen objek pajak bumi dan bangunan sektor perdesaan dan perkotaan.

3.2 Bagian Logika Aplikasi

Pada bagian ini nantinya akan terdiri dari diagram struktur dan diagram perilaku dari aplikasi, bagaimana masing-masing komponen saling bertukar informasi, bagaimana alur data dari basis data ke bagian tampilan dan sebaliknya dimodelkan pada bagian ini.

Bagian ini bisa disebut inti dari aplikasi, karena bagian ini yang nantinya mengontrol transfer data antar komponen dan lapisan.

3.3 Bagian Tampilan Aplikasi

Istilah lainnya biasa dikenal dengan antar muka pengguna, bagian ini yang nantinya akan menjadi desain atau model dari tampilan yang berhadapan langsung dengan pengguna. Bagian ini yang nantinya mengumpulkan informasi untuk disampaikan kepada bagian logika aplikasi untuk diproses, bagian ini pula yang nantinya menampilkan informasi yang diproses oleh sistem untuk dapat dibaca dan dipahami oleh pengguna.

4 PERTIMBANGAN KHUSUS KINERJA SISTEM

Karena sistem datanya terpusat, yaitu tersimpan pada peladen basis data Oracle, maka perawatan atau pemeliharaan mutlak dilakukan pada sisi peladen sistem basis data yang diakibatkan dari peningkatan jumlah data transaksi pembayaran yang selalu bertambah dari sisi ukuran dari hari ke hari.

Biasanya akan dilakukan pembersihan terhadap berkas *log*, yaitu berkas yang digunakan oleh sistem basis data Oracle untuk mencatatkan kegiatan atau aktivitas yang telah dilakukannya selama beroperasi melayani permintaan data. Pembersihan ini dilakukan secara rutin karena transaksi yang terjadi pun rutin dilakukan dalam periode harian.

Selain peladen sistem basis data, peladen aplikasi *web* pun perlu mendapatkan perhatian pada sisi perawatan atau pemeliharaan, dimana peladen aplikasi *web* pun memiliki berkas *log* tersendiri yang perlu dibersihkan secara rutin karena menyimpan transaksi antar komputer klien dan peladen.

Hal lain yang perlu dilakukan sebagai langkah pengamanan data yaitu dilakukannya duplikasi data (*backup data*) sehingga apabila terjadi hal-hal yang

tidak diinginkan, data dapat langsung dikembalikan atau dipulihkan (*recovery*). Ini pun sebaiknya dilakukan secara rutin, sebaiknya dalam periode harian.

5 HASIL PEMODELAN

Hasil pemodelan untuk sistem informasi pembayaran pajak bumi dan bangunan sektor perdesaan dan perkotaan nantinya akan dibagi menjadi beberapa diagram mendasar pada beberapa bagian dari arsitektur sistem, berikut rincian mengenai hasil pemodelan tersebut :

5.1 Bagian Basis Data

Basis data yang berhubungan dengan aplikasi ini sebetulnya menggunakan sistem basis data yang telah digunakan untuk pengelolaan Pajak Bumi dan Bangunan sektor Perdesaan dan Perkotaan, namun hanya beberapa tabel saja yang diakses oleh aplikasi ini, diagram tabel yang berhubungan dengan aplikasi sistem informasi pembayaran pajak bumi dan bangunan sektor perdesaan dan perkotaan ini adalah seperti gambar 1 berikut ini :

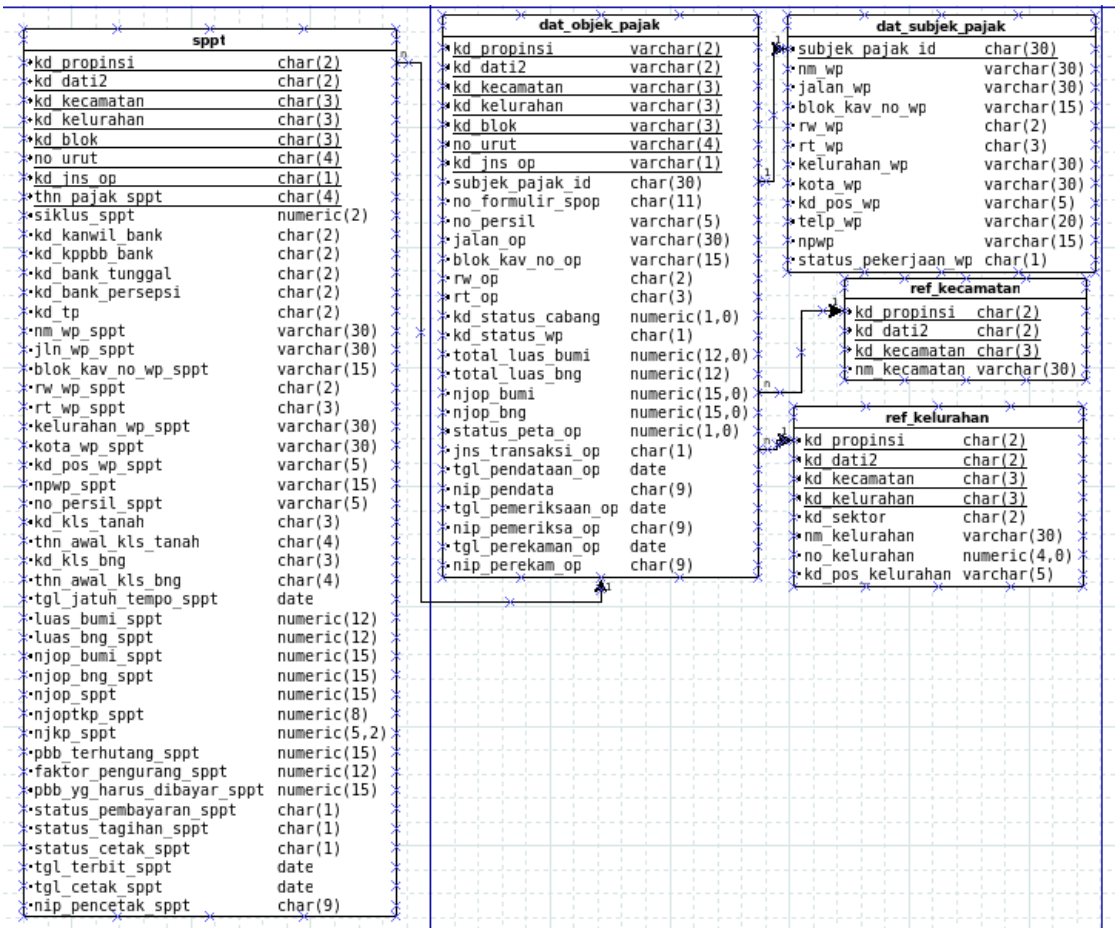

Figure 1: Diagram Tabel Yang Digunakan

5.2 Bagian Logika Aplikasi

Ada beberapa pemodelan logika aplikasi di dunia teknologi informasi, jika pengembangan dilakukan dengan bahasa pemrograman terstruktur, maka akan menggunakan *flowchart* atau diagram alir sebagai alat untuk pemodelan, karena yang akan digunakan sekarang adalah bahasa pemrograman Java yang menggunakan metodologi orientasi objek, maka digunakan *Unified Modeling Language* (UML) yang akan menggambarkan struktur sistem dari awal sampai akhir. Berikut adalah pemodelan atau diagram yang menggambarkan bagian sistem informasi pemba-

yaran Pajak Bumi dan Bangunan Perdesaan dan Perkotaan bekerja.

5.2.1 Diagram *Use Case*

Diagram *use-case* dari keseluruhan sistem yang akan dibangun adalah seperti gambar 2 ini :

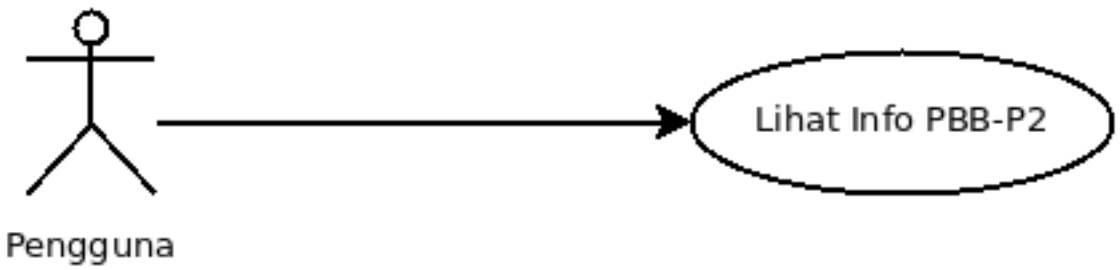

Figure 2: Diagram *Use-Case*

Sistem aplikasi hanya akan menampilkan sebuah informasi saja kepada pengguna yaitu berupa informasi besaran tagihan pajak yang terhutang dan status pembayaran pajaknya.

5.2.2 Diagram *Class*

Diagram *class* pada sistem aplikasi ini akan terbagi menjadi 2 (dua) bagian, yaitu bagian ujung depan dan bagian ujung belakang. Diagram *class* untuk bagian ujung belakang adalah seperti pada gambar 3 dan 4 berikut ini :

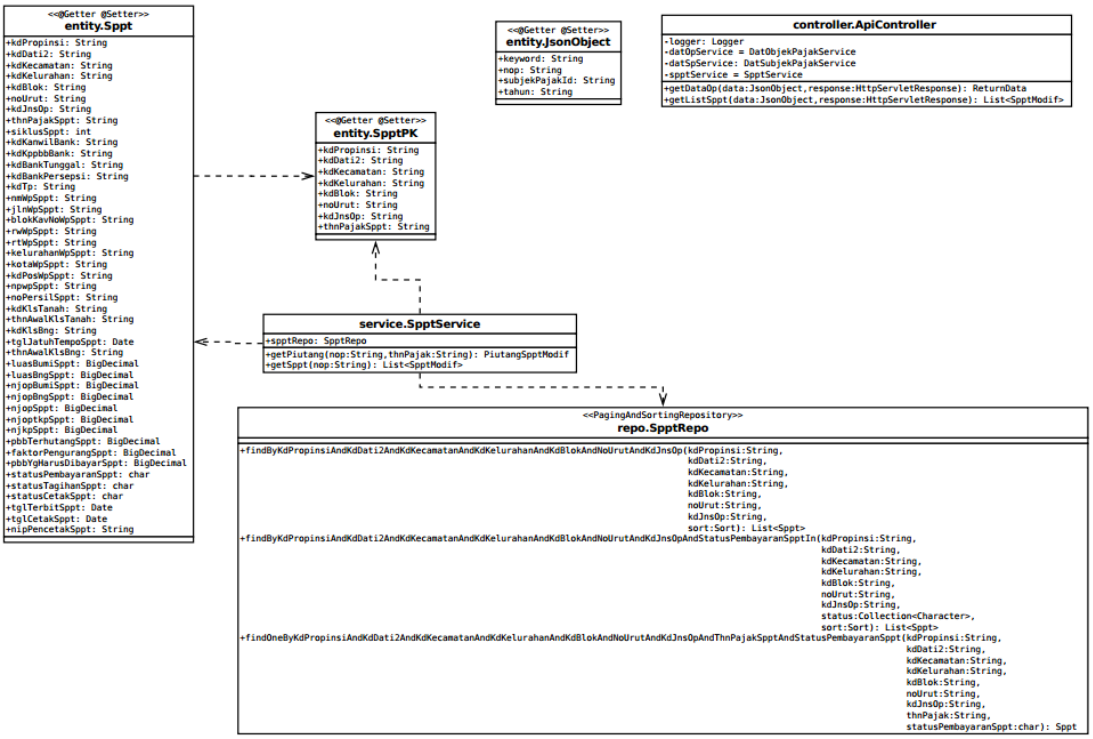

Figure 3: Diagram *Class* Untuk Ujung Belakang Bagian 1

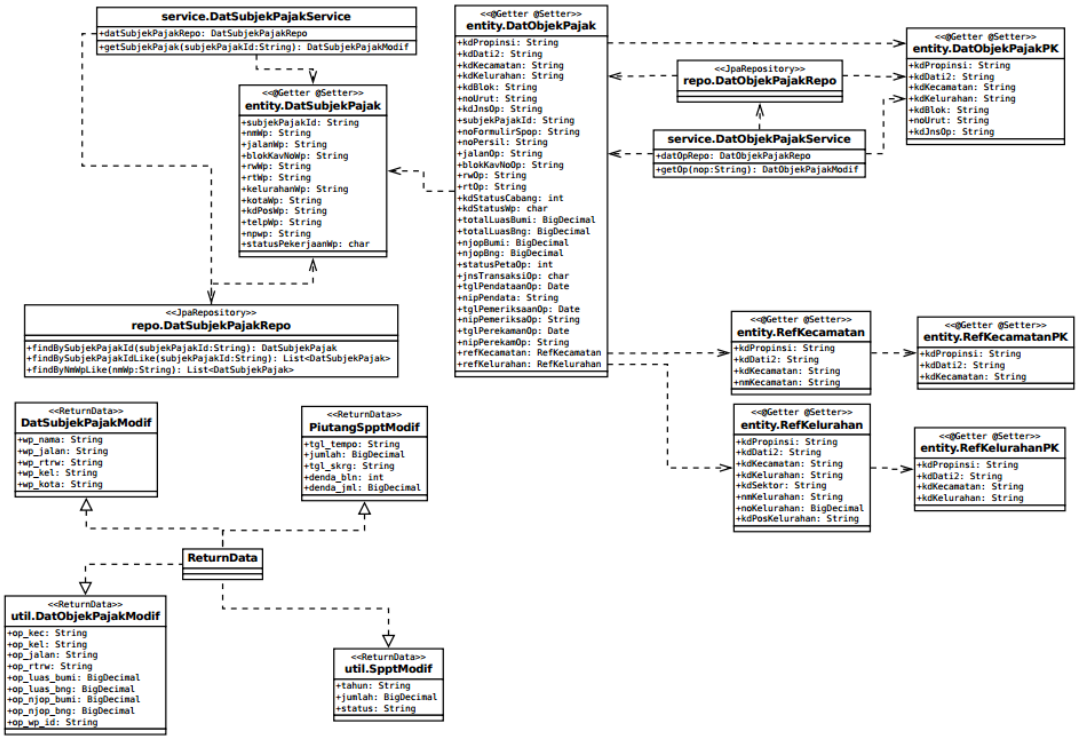

Figure 4: Diagram *Class* Untuk Ujung Belakang Bagian 2

Pada diagram *class* yang pertama, ada kelompok kelas dengan nama yang mirip yaitu `Sppt`, karena implementasinya menggunakan Spring Data JPA, maka membutuhkan beberapa kelas atau *interface* untuk mengelola data yang berasal dari sistem basis data.

Kelas dan *interface* yang berhubungan dengan tabel `SPPT` ini adalah seperti berikut :

- Kelas `Sppt`, digunakan untuk melakukan pemetaan atribut pada tabel `SPPT` pada sistem basis data, isi atributnya akan mirip dengan isi atribut pada tabelnya.

- Kelas `SpptPK`, digunakan untuk mendeklarasikan *primary key* atau kunci utama dari tabel `SPPT`, nantinya kelas ini akan digunakan pada kelas `Sppt`

untuk memetakan *field-field* yang menjadi *primary key*.

- *Interface* `SpptRepo` adalah deklarasi yang fungsinya untuk melakukan operasi terhadap tabel `SPPT` di sistem basis data melalui kelas entitas yang berkaitan, dalam hal ini adalah kelas `Sppt`.

- Kelas `SpptService` digunakan untuk mengelola data atau manipulasi data yang datang dari sistem basis data atau yang akan disimpan ke dalam basis data.

Pada diagram *class* bagian pertama ini pun ada 2 (dua) kelas yang tidak berhubungan langsung dengan kelas `Sppt` yaitu kelas `JsonObject` yang sebetulnya digunakan untuk pemetaan dari objek JSON yang dikirimkan oleh klien, diubah atau dikonversi menjadi objek Java secara otomatis dengan menggunakan pustaka Jackson. Kemudian kelas yang lain adalah `ApiController` yang isinya adalah pemetaan URL yang dapat di*request* oleh klien untuk memperoleh data, segala proses yang berhubungan dengan *request* data akan dilakukan pada kelas ini.

Pada diagram *class* bagian kedua akan berisi beberapa kelas dan *interface* yang berhubungan dengan tabel `DAT_OBJEK_PAJAK` beserta beberapa tabel yang berhubungan dengan tabel tersebut.

Adapun kelas-kelas dan *interface* yang ada pada diagram *class* ini yang berhubungan dengan tabel `DAT_OBJEK_PAJAK` yaitu :

- Kelas `DatSubjekPajakService`, kelas ini bertugas untuk melakukan manipulasi atau pengolahan data yang berasal dari sistem basis data, atau akan disimpan ke sistem basis data.

- Kelas `DatSubjekPajak`, kelas ini berfungsi sebagai kelas pemetaan untuk tabel `DAT_SUBJEK_PAJAK`, maka dari itu isi properti dari kelas ini akan mirip seperti isi properti dari tabel `DAT_SUBJEK_PAJAK`.

- *Interface* `DatSubjekPajakRepo`, *interface* ini berfungsi untuk melakukan operasi terhadap isi data pada tabel `DAT_SUBJEK_PAJAK` seperti simpan data, ubah data, hapus data, ambil data.

- Kelas `DatObjekPajak`, kelas ini berfungsi untuk melakukan pemetaan terhadap tabel `DAT_OBJEK_PAJAK`, seperti kelas `DatSubjekPajak`, pada kelas ini pun isi propertinya akan mirip seperti pada tabel `DAT_OBJEK_PAJAK`.

- *Interface* `DatObjekPajakRepo`, seperti *interface repository* lainnya, bertugas untuk melakukan manipulasi data pada tabel `DAT_OBJEK_PAJAK` pada sistem basis data.

- Kelas `DatObjekPajakPK`, kelas ini digunakan untuk memetakan *primary key* dari tabel `DAT_OBJEK_PAJAK` yang digunakan oleh kelas `DatObjekPajak`.

- Kelas `DatObjekPajakService`, kelas ini digunakan untuk mengadaptasikan atau mengolah data dari sistem basis data ke antar muka pengguna, atau sebaliknya, dari antar muka pengguna ke sistem basis data.

- Kelas `RefKecamatan`, kelas ini akan di rujuk oleh kelas *DatObjekPajak* untuk menampilkan nama wilayah Kecamatan dimana objek berada.

- Kelas `RefKecamatanPK`, kelas ini digunakan sebagai kelas pemetaan untuk *primary key* dari tabel `RefKecamatan`.

- Kelas `RefKelurahan`, kelas ini digunakan untuk memetakan tabel `REF_KELURAHAN` yang digunakan untuk mereferensikan nama wilayah Kelurahan untuk objek pajak terpilih.

- Kelas `RefKelurahanPK`, digunakan untuk memetakan *primary key* dari tabel `REF_KELURAHAN` yang akan digunakan pada kelas `RefKelurahan`.

Kelas dan *interface* lain akan berhubungan dengan respon data atau format pemberian data ke klien. Kelas dan *interface* tersebut adalah seperti berikut ini :

- *Interface* `ReturnData`, *interface* ini tidak memiliki isi apapun, hanya untuk menyeragamkan deklarasi kelas-kelas untuk respon data ke klien.

- Kelas *DatSubjekPajakModif*, kelas ini akan mengembalikan nilai-nilai atau informasi mengenai data subjek pajak dari basis data berdasarkan Nomor Objek Pajak yang telah diberikan, hanya beberapa informasi saja yang di-ikutsertakan dalam blok informasi ini sesuai dengan nama propertinya.

- Kelas *PiutangSpptModif*, kelas ini akan membawa informasi mengenai data piutang pembayaran pajak bumi dan bangunan sektor perdesaan dan perkotaan sesuai dengan Nomor Objek Pajak yang diberikan

- Kelas *DatObjekPajakModif*, kelas ini akan membawa informasi mengenai data informasi objek pajak secara umum seperti tertuang pada propertinya. Tentu saja nilai yang dibawa berdasarkan Nomor Objek Pajak yang diberikan.

- Kelas *SpptModif*, kelas ini akan membawa informasi status tagihan pajak berdasarkan tahun pajaknya.

Demikianlah isi seluruh kelas dan *interface* yang akan membangun aplikasi pada bagian ujung-belakang (*backend*) agar berjalan sebagaimana mestinya.

Diagram *class* untuk bagian ujung-depan (*frontend*) terlihat lebih sederhana lagi, gambar 5 menunjukkan diagram tersebut beserta hubungan keterkaitan antar kelasnya :

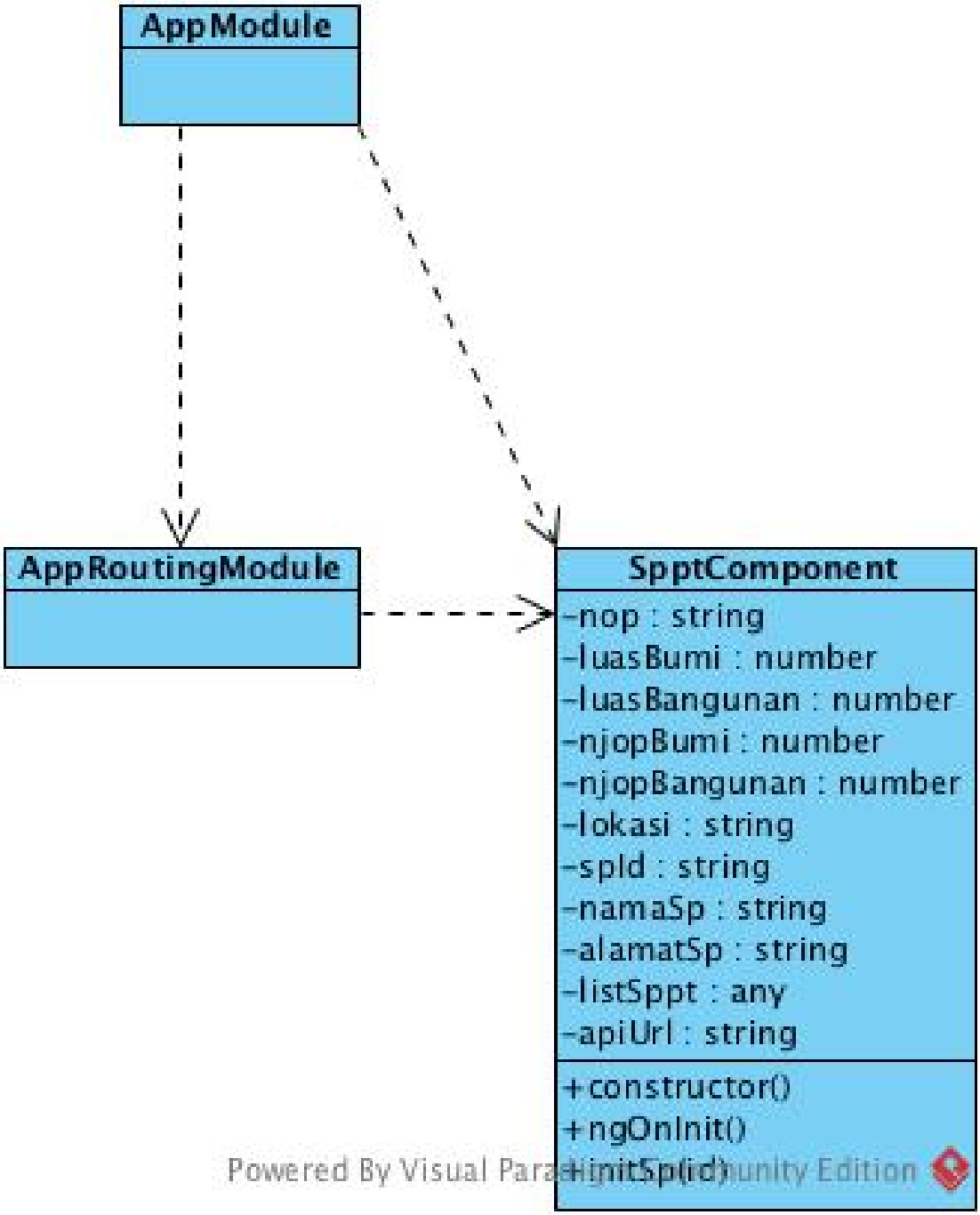

Figure 5: Diagram *Class* Bagian Ujung-Depan (*frontend*)

Pada bagian ujung depan (*frontend*), aplikasi akan berakar pada kelas

`AppModule`, kelas `AppRoutingModule` yang akan mengatur bagaimana tiap komponen kelas (dalam hal ini halaman) akan berganti-ganti sesuai dengan kejadian yang diinginkan oleh pengguna.

Kelas `HomeComponent` dan `SpptComponent` adalah 2 (dua) kelas yang berhubungan langsung dengan tampilan aplikasi (*view*), keduanya akan dikontrol langsung oleh kelas `AppRoutingModule` bilamana halaman yang ditampilkan terlebih dahulu, apakah `HomeComponent` atau `SpptComponent`.

Kelas `HomeComponent` sebetulnya kelas yang bertugas menjadi tampilan utama aplikasi, artinya begitu ada *browser* yang melakukan akses, halaman dari kelas `HomeComponent` ini akan tampil terlebih dahulu menyambut pengguna.

Kelas `SpptComponent` nanti akan menampilkan informasi yang diminta oleh pengguna berdasarkan Nomor Objek Pajak yang telah dikirimkan melalui formulir atau komponen yang tersedia.

5.2.3 Diagram *Package*

Diagram ini akan menunjukan struktur desain sistem pada level *package*, diagramnya adalah seperti pada gambar 6 berikut ini :

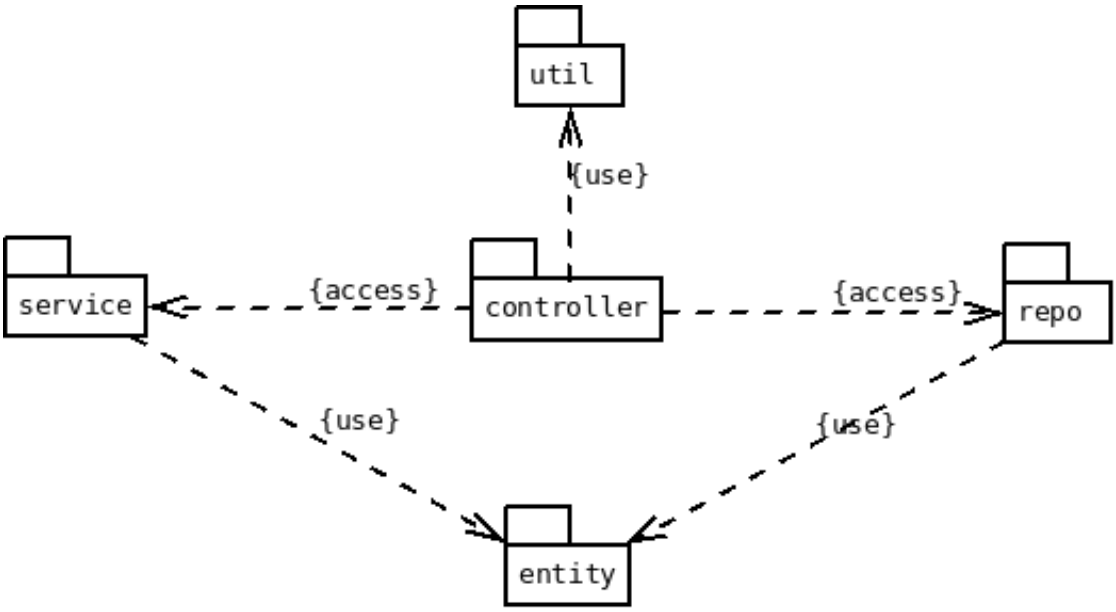

Figure 6: Diagram *Package*

Package `controller` akan menggunakan paket `util` sebagai respon data ke klien dengan cara mengakses paket `service` dan paket `repo` berdasarkan Nomor Objek Pajak yang dikirimkan oleh klien, kemudian paket `service` dan paket `repo` akan menggunakan paket `entity` karena Spring Data JPA dalam melakukan akses data ke sistem basis data akan selalu mengembalikan nilai kedalam kelas-kelas pemetaan pada paket `entity` ini.

5.2.4 Diagram *Component*

Diagram ini memberikan gambaran hubungan antar komponen, komponen mana yang membutuhkan data dan komponen mana yang memberikan data akan terlihat jelas pada diagram komponen ini. Berikut adalah diagram komponen yang membentuk sistem ini :

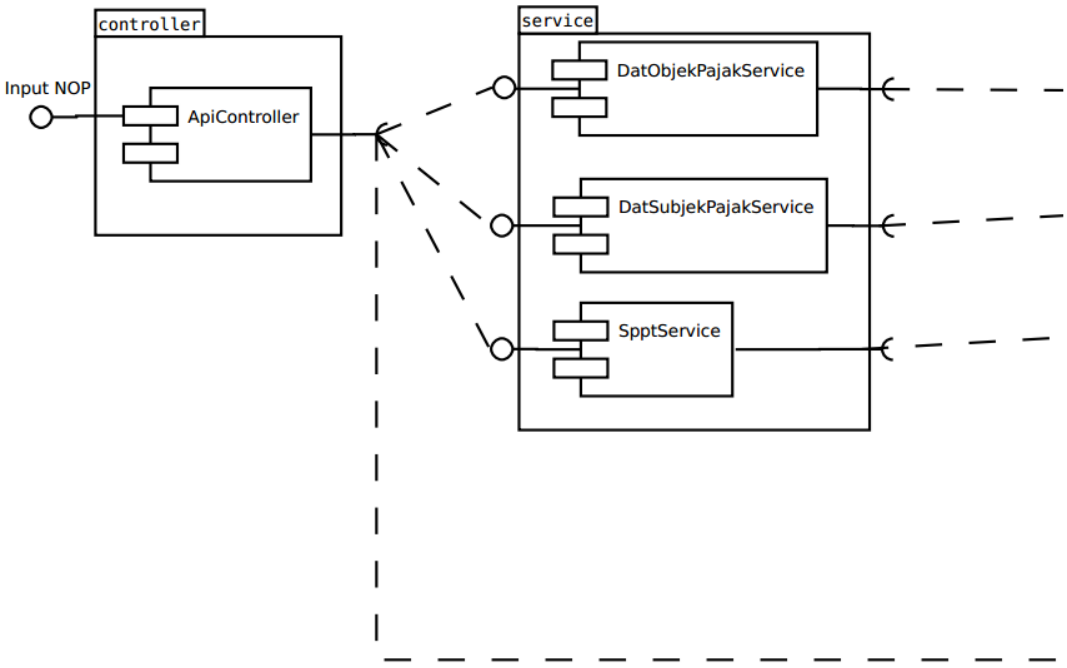

Figure 7: Diagram *Component* Bagian 1

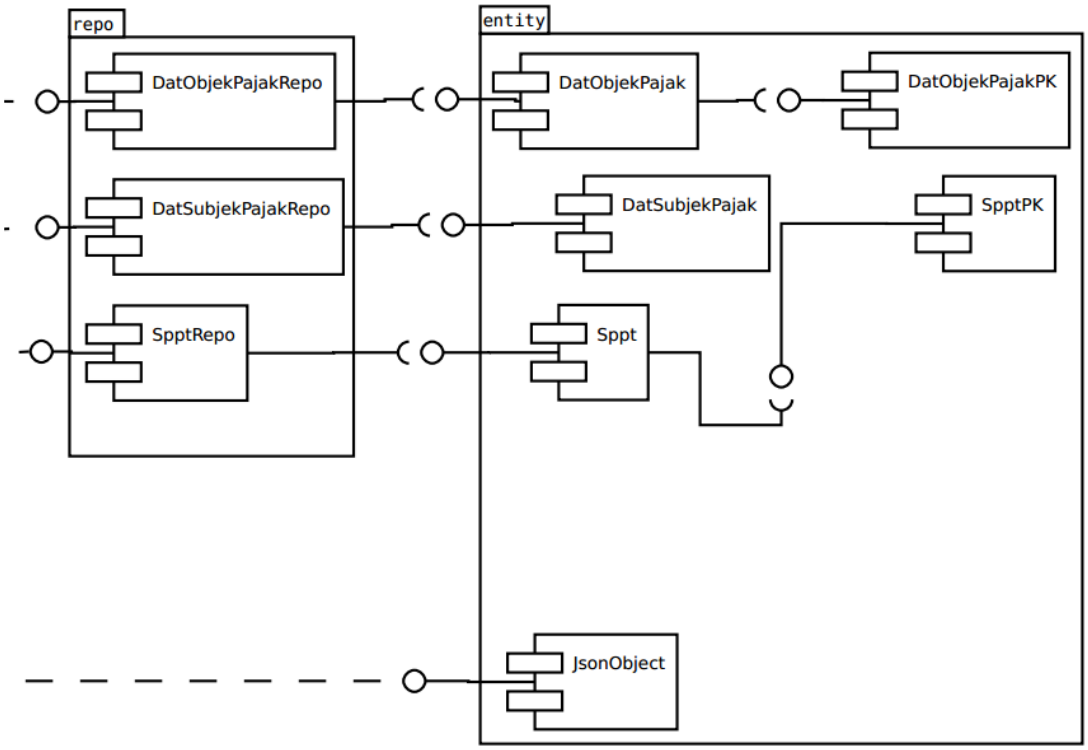

Figure 8: Diagram *Component* Bagian 2

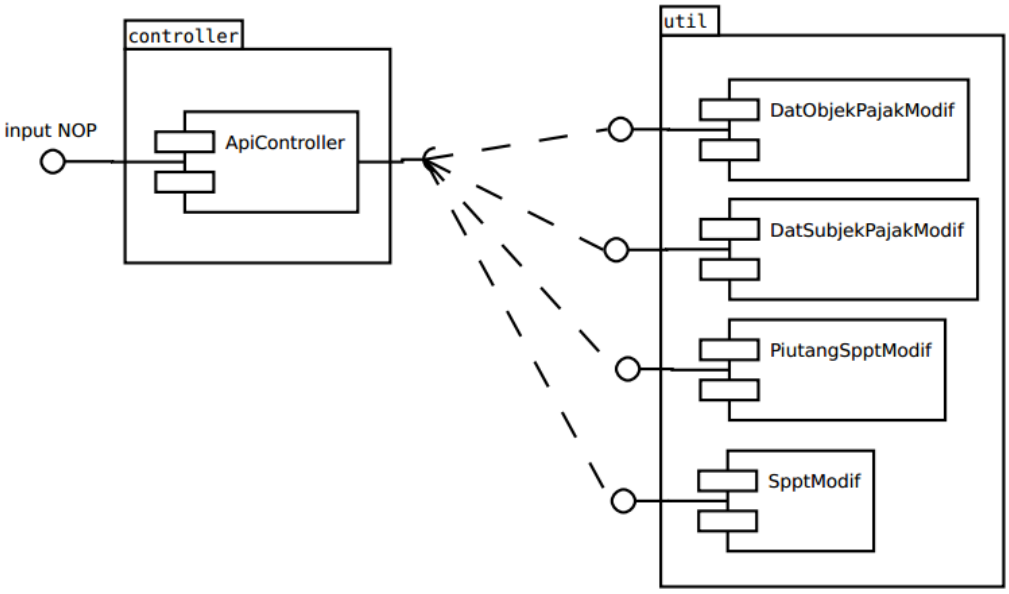

Figure 9: Diagram *Component* Bagian 3

Diagram tersebut berisi komponen yang membangun sistem ini menjadi utuh, begitu ada masukkan data yang terjadi terhadap `ApiController`, maka `ApiController` akan menghubungi salah satu atau keseluruhan *service* sesuai dengan *request* yang diterima. Data yang diterima oleh `ApiController` sebetulnya akan berbentuk JSON, namun akan diterjemahkan otomatis oleh pustaka Jackson ke dalam kelas `JsonObject` seperti terlihat pada gambar 8.

Kemudian tiap *service* akan melakukan pemanggilan data terhadap *repository*-nya masing-masing yang kemudian akan dikembalikan dalam bentuk objek sesuai tabel yang diaksesnya seperti pada gambar 7 dan 8.

Hasil yang dikembalikan adalah beberapa objek dari kelas pada paket `util` seperti disebutkan pada gambar 9.

Di sisi lain, diagram *component* untuk bagian ujung-depan (*frontend*) ditunjukkan seperti pada gambar 5.2.4 berikut ini :

Figure 10: Diagram *Component* Untuk Ujung-Depan (*frontend*)

Pada `AppComponent` sudah terdapat formulir untuk memasukkan Nomor Objek Pajak, yang apabila diproses maka `SpptComponent` akan melakukan *request* data ke peladen API untuk memperoleh data berdasarkan Nomor Objek Pajak yang telah dimasukkan oleh pengguna, hasil dari *request* ini akan ditampilkan pada `SpptComponent` pula.

5.2.5 Diagram *Deployment*

Diagram ini menunjukkan arsitektur dari sistem pada saat didistribusikan dari mesin tempat untuk mengembangan dan uji coba, ke mesin produksi tempat aplikasi siap untuk melayani pengguna aslinya.

Diagram ini digambarkan seperti pada gambar 5.2.5 berikut :

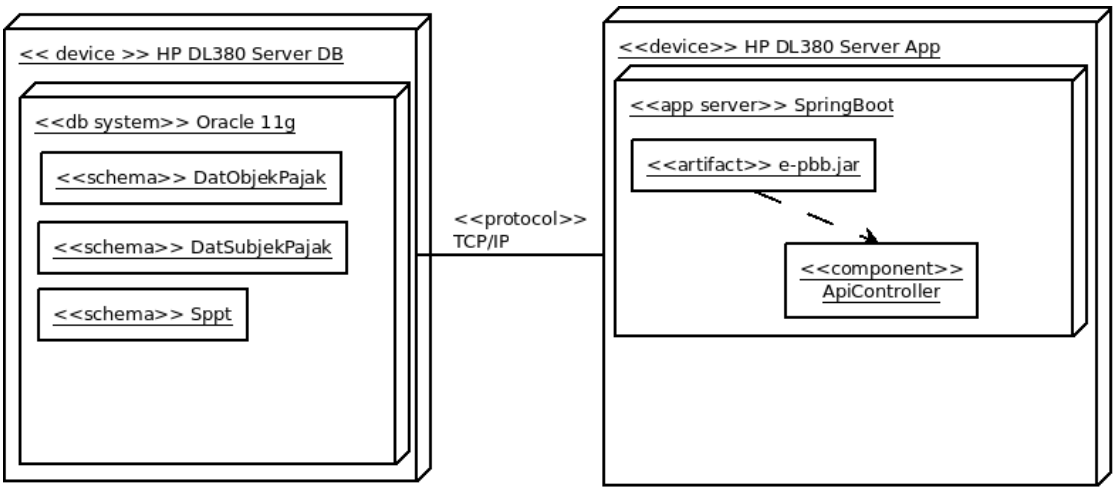

Figure 11: Diagram *Deployment*

Pada diagram tersebut ditunjukkan bahwa sistem aplikasi ini menggunakan 2 (dua) peladen, yang pertama untuk sistem basis data seperti ditunjukkan pada gambar yang memiliki sistem basis data (perangkat / *device* di sebelah kiri), dan yang kedua adalah peladen aplikasi dengan Tomcat dan lingkungan berada dalam satu paket pada Springboot.

5.2.6 Diagram *Communication*

Diagram ini menggambarkan interaksi antar objek yang disertai urutan komunikasi dalam bentuk bagan yang bebas.

Untuk bagian ujung-depan (*frontend*), diagram *communication* diperlihatkan seperti pada gambar 12 berikut ini :

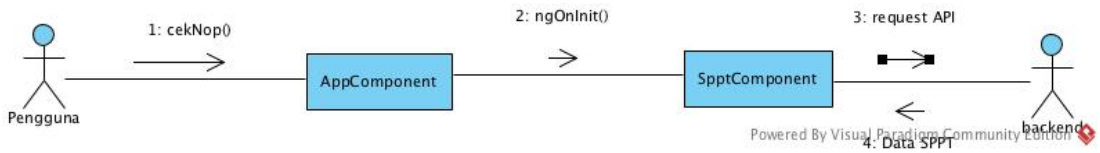

Figure 12: Diagram *Communication* Untuk Bagian Ujung-Depan (*frontend*)

Alurnya adalah dari pengguna akan memasukkan Nomor Objek Pajak (NOP) pada kolom yang disediakan pada `AppComponent`, kemudian komponen ini akan memanggil fungsi `cekNop()` dan mengaktifkan `SpptComponent` yang secara otomatis akan memanggil fungsi `ngOnInit()`.

Selanjutnya bagian ujung-depan (*frontend*) akan melakukan *request* ke ujung-belakang (*backend*) yang kemudian ujung belakang akan memberikan data yang diminta dan ditampilkan pada `SpptComponent`.

Diagram *communication* yang membentuk sistem informasi pajak bumi dan bangunan sektor perdesaan dan perkotaan pada bagian ujung-belakang (*backend*) ini adalah seperti pada gambar 13 dan 14 berikut ini :

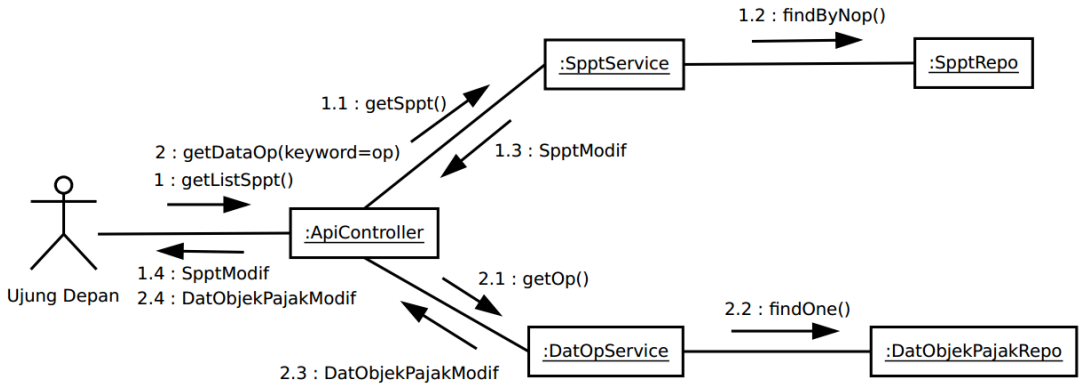

Figure 13: Diagram *Communication* Bagian 1

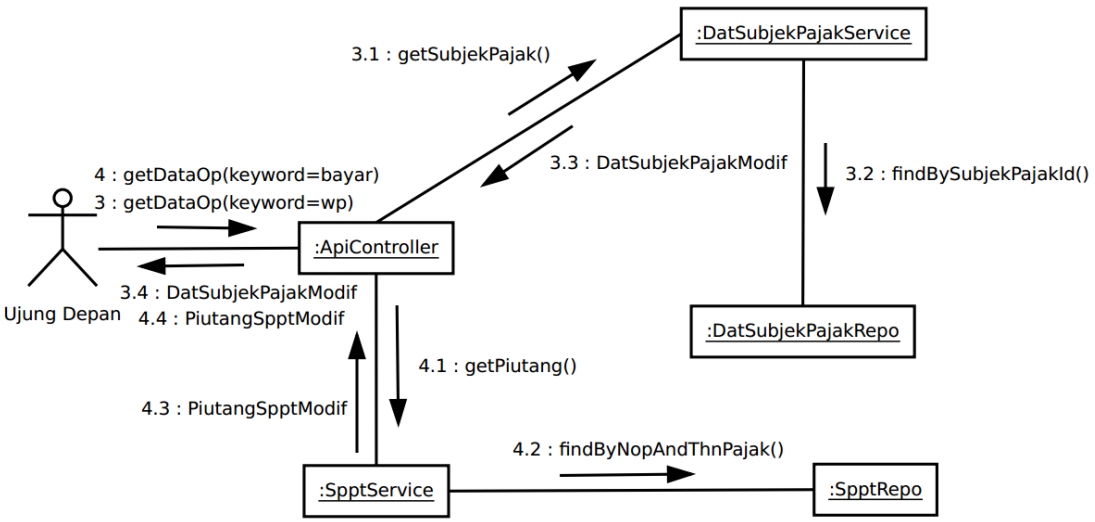

Figure 14: Diagram *Communication* Bagian 2

Nomor urut 1 (satu) dan 2 (dua) pada saat *Ujung Depan* (*front-end*) berkomunikasi dengan peladen API (dalam hal ini komponen `ApiController`) bukan berarti urutan dari alur komunikasi melainkan identitas dari skenario. Namun untuk simbol atau angka seperti 1.1 (satu titik satu), 1.2 (satu titik dua), dan seterusnya adalah urutan komunikasi dari skenario 1 (satu) dari langkah pertama

sampai langkah ke-n.

Jadi untuk skenario yang ke-1 (satu), bagian *ujung depan* (*front-end*) akan memanggil *method* `getListSppt()` milik kelas `ApiController`, proses dari skenario ini akan memanggil *method* `getSppt()` milik kelas `SpptService` (pada langkah 1.1), proses selanjutnya akan memanggil *method* `findByNop()` milik kelas `SpptRepo` (pada langkah 1.2) yang akhirnya akan mengembalikan ke kelas *ApiController* dan ke *ujung depan* (*front-end*) berupa objek dari kelas `SpptModif` (pada langkah 1.3 dan 1.4).

Pada skenario ke-2, *ujung depan* (*front-end*) akan memanggil *method* `getDataOp()` milik kelas `ApiController` dengan parameter `keyword` berisi teks `op`, prosesnya pertama akan memanggil *method* `getOp()` milik kelas `DatOpService` (seperti pada langkah 2.1), kemudian proses berlanjut dengan memanggil fungsi `findOne()` dari objek kelas `DatObjekPajakRepo`, hasil dari pemanggilan *method* `findOne()` ini akan dikonversi ke dalam objek kelas `DatObjekPajakModif` yang pada akhirnya akan dikembalikan ke aplikasi bagian *ujung depan* (*front-end*) dalam bentuk JSON.

Pada skenario ke-3 di gambar 14, aplikasi bagian *ujung depan* (*front-end*) akan melakukan *request* ke *method* `getDataOp()` dengan parameter `keyword` berisi teks `wp`. Langkah selanjutnya adalah memanggil *method* `getSubjekPajak()` milik kelas `DatSubjekPajakService`, yang proses kemudian akan memanggil *method* `findBySubjekPajakId()`, hasilnya kemudian akan disesuaikan masuk ke dalam kelas `DatSubjekPajakModif` yang dikembalikan ke bagian *ujung depan* (*front-end*).

Skenario ke-4 sama saja skenario ke-3 dan ke-2, akan memanggil *method* `getDataOp()` dengan parameter `keyword` berisi teks `bayar`. Langkah selanjutnya adalah memanggil *method* `getPiutang()` milik kelas `SpptService` yang kemudian akan mengambil data ke sistem basis data dengan memanggil *method* `findByNopAndThnPajak()` milik kelas `SpptRepo`. Hasil dari proses ini akan

mengembalikan ke bagian *ujung depan* (*front-end*) berupa objek dari kelas `PiutangSpptModif` dalam bentuk JSON.

5.2.7 Diagram *Activity*

Diagram ini masuk dalam kategori diagram *behavior* yang menunjukkan alur kontrol atau alur objek yang dipertegas dalam urutan aktivitas dan kondisi pada alur yang terjadi.

Diagram *activity* untuk bagian ujung-depan (*frontend*) adalah seperti terlihat pada gambar 15 berikut ini :

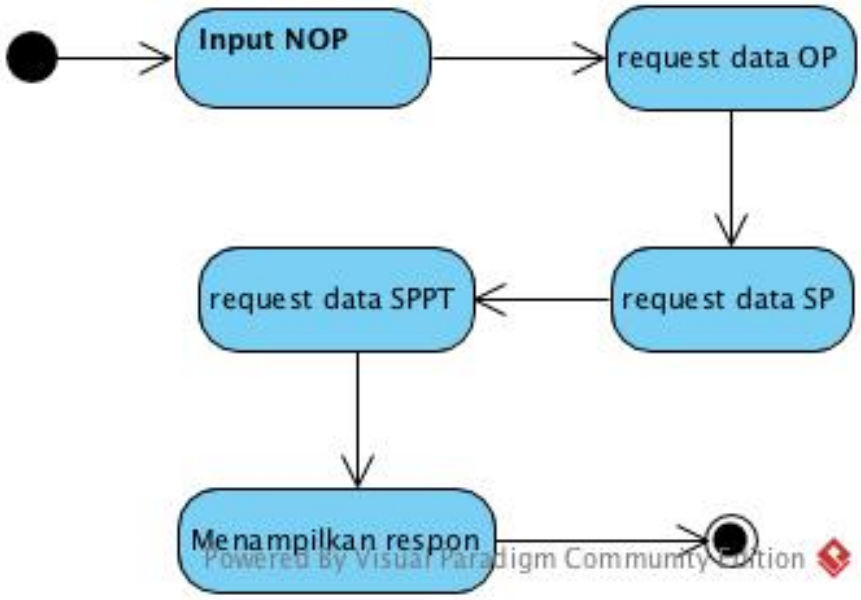

Figure 15: Diagram *Activity* Untuk Bagian Ujung-Depan (*frontend*)

Terlihat pada diagram tersebut bahwa bagian ujung-depan (*frontend*) akan melakukan 3 (tiga) kali *request* atau permintaan ke peladen API, kemudian menampilkan hasil yang diberikan oleh peladen ke jendela *browser*.

Pada bagian ujung-belakang (*backend*), diagram *activity* ini akan terbagi berdasarkan skenario yang telah terbentuk pada Diagram *Communication* se-

belumnya.

Pada skenario pertama, adalah aktivitas yang terjadi ketika ada *request* daftar tagihan pajak bumi dan bangunan sektor perdesaan dan perkotaan dari klien, diagram *activity* untuk skenario ini adalah seperti pada gambar 16 berikut ini :

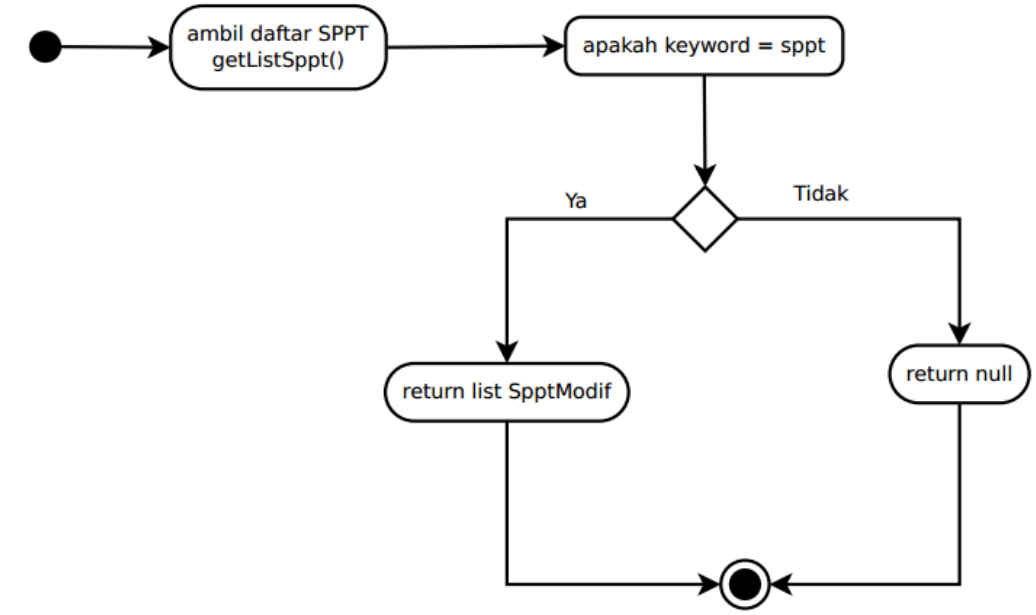

Figure 16: Diagram *Activity* Untuk Ambil Daftar Tagihan

Diagram tersebut menunjukkan apabila data yang diminta oleh klien tidak ada pada sistem basis data, maka sistem akan mengembalikan nilai `null` atau kosong.

Skenario yang kedua terjadi ketika ada *request* data objek pajak dari klien. Diagram *activity* untuk skenario kedua ini seperti terlihat pada gambar 17 berikut ini :

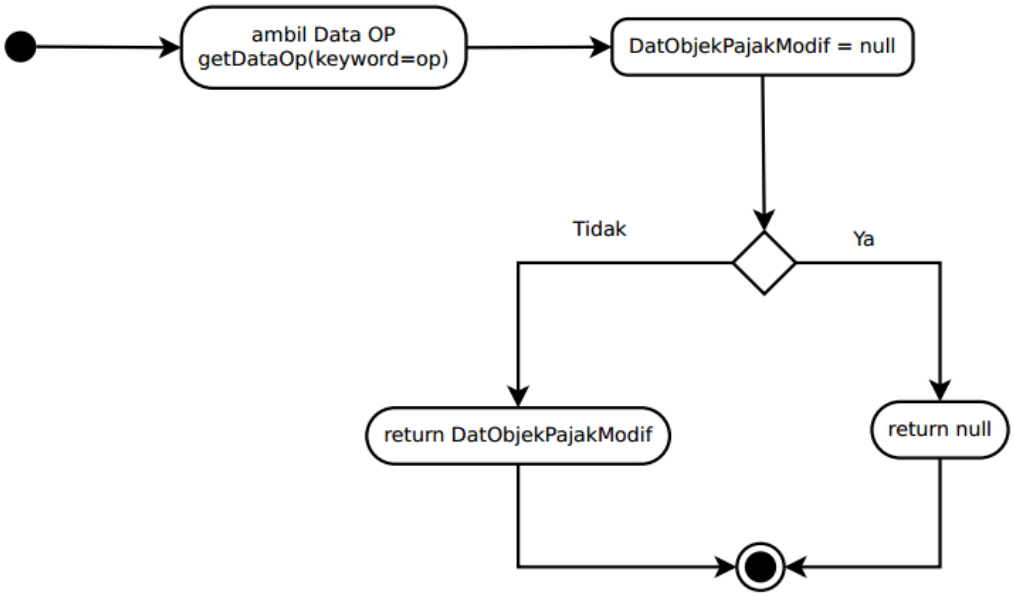

Figure 17: Diagram *Activity* Skenario Kedua

Skenario berikutnya adalah skenario ketiga, yang terjadi ketika ada *request* data wajib pajak dari klien. Diagram *activity* untuk skenario ini adalah seperti pada gambar 18 berikut ini :

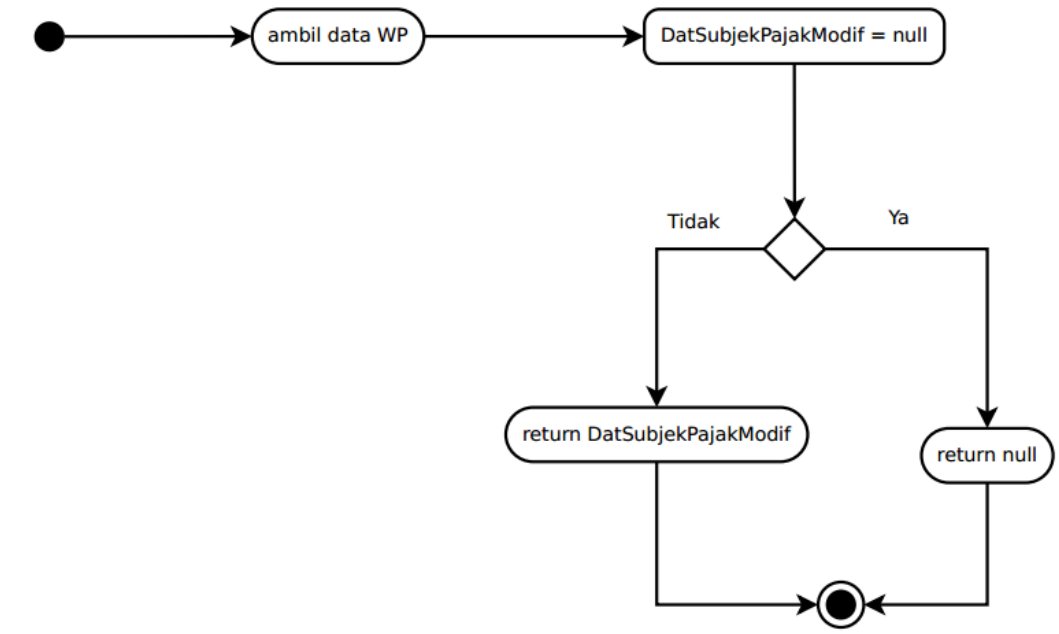

Figure 18: Diagram *Activity* Untuk Skenario Ketiga

Skenario keempat terjadi ketika ada *request* data pembayaran dari klien. Diagram *activity* untuk skenario ini adalah seperti pada gambar 19 berikut :

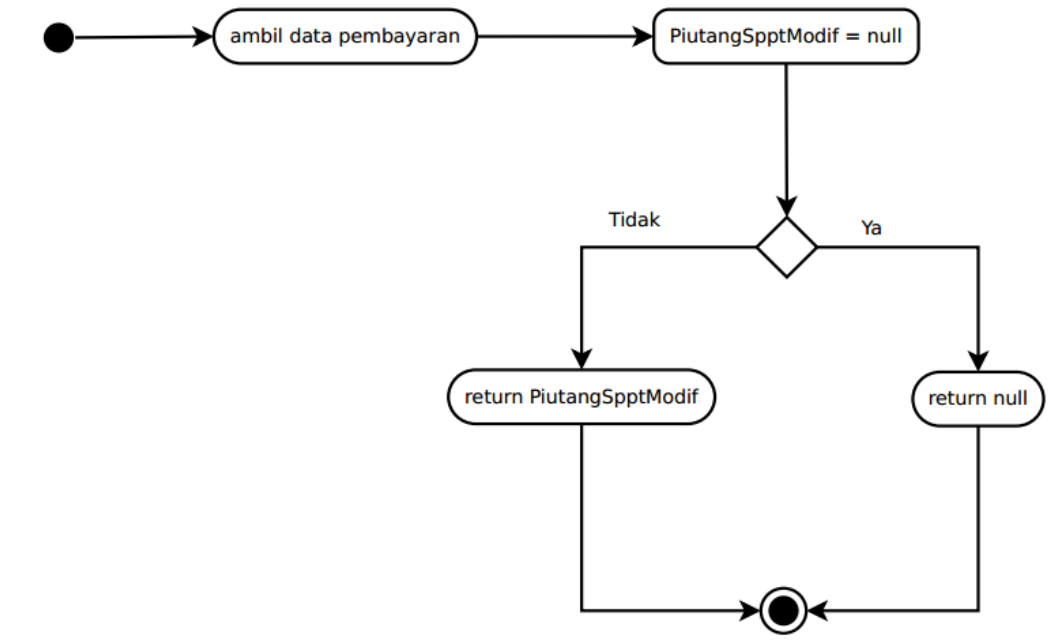

Figure 19: Diagram *Activity* Untuk Skenario Keempat

5.2.8 Diagram *Sequence*

Diagram ini akan menggambarkan alur pertukaran pesan dari beberapa objek pada rentang siklus hidupnya.

Untuk bagian ujung depan (*frontend*), diagram *sequence* yang menggambarkan alur kontrol adalah seperti pada gambar 20 berikut ini :

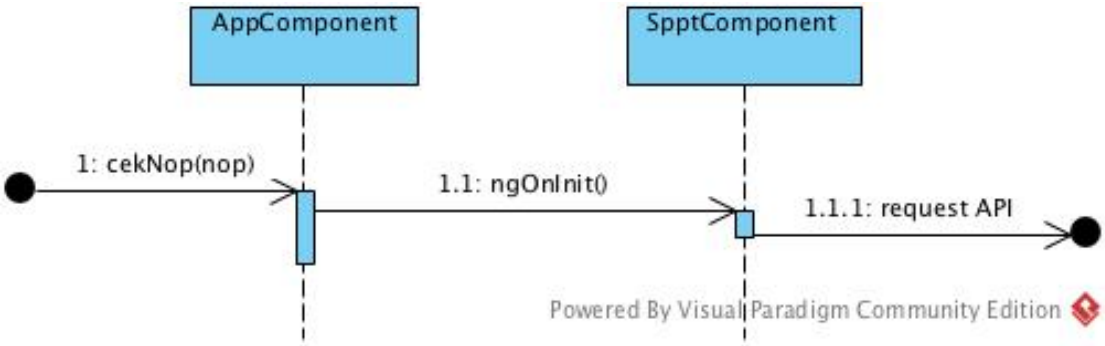

Figure 20: Diagram *Sequence* Untuk Bagian Ujung-Depan (*frontend*)

Prosesnya terlihat cukup sederhana, yaitu dari saat pengguna memasukkan Nomor Objek Pajak pada komponen yang tersedia, kemudian begitu pengguna melakukan klik pada tombol yang disediakan, maka otomatis akan memanggil fungsi `cekNop()` dengan parameter berupa Nomor Objek Pajak (NOP), kemudian aplikasi akan membuka / memanggil `SpptComponent` yang di dalamnya kemudian melakukan *request* terhadap API pada peladen bagian ujung belakang (*backend*).

Pada bagian ujung-belakang (*backend*), diagram ini pun terbentuk dari skenario-skenario yang terjadi dari diagram *communication* sebelumnya.

Pada skenario pertama, sistem akan memberikan daftar tagihan pajak bumi dan bangunan sektor perdesaan dan perkotaan untuk seluruh wajib pajak berdasarkan nomor objek pajak yang diminta.

Diagram *sequence* untuk skenario pertama ini seperti pada gambar 5.2.8 berikut ini :

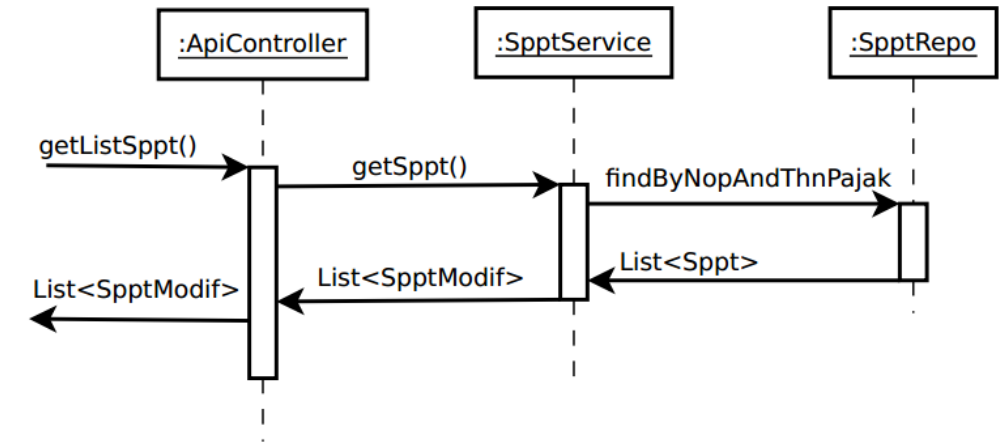

Figure 21: Diagram *Sequence* Untuk Skenario Pertama

Untuk skenario kedua, sistem akan memberikan informasi mengenai objek pajak berdasarkan nomor objek pajak yang diminta. Diagram *sequence* untuk skenario ini adalah seperti pada gambar 22 berikut ini :

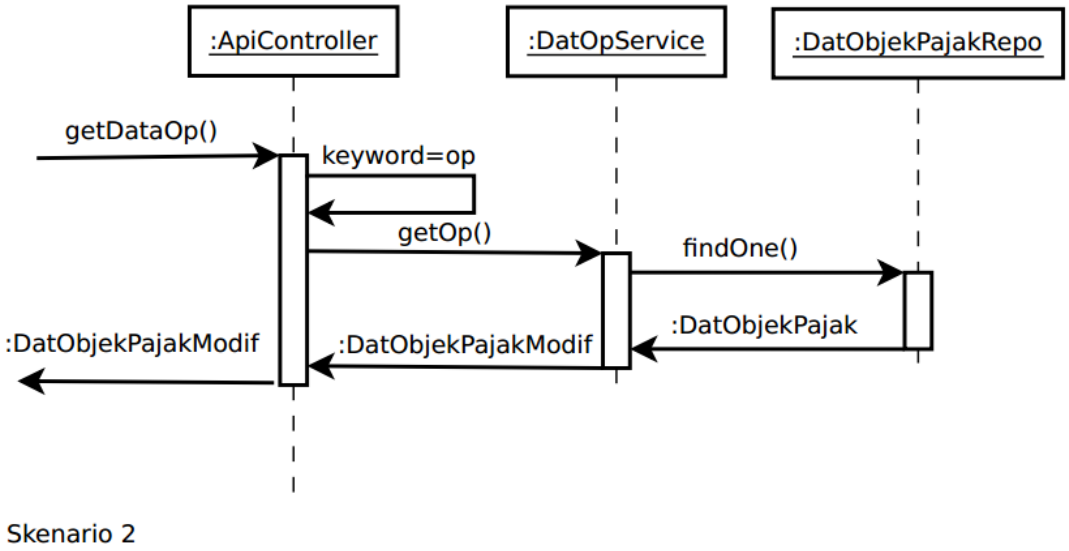

Figure 22: Diagram *Sequence* Untuk Skenario Kedua

Untuk skenario ketiga, sistem akan memberikan informasi mengenai wajib pajak berdasarkan nomor objek pajak yang diminta. Diagram *sequence* untuk skenario ini adalah seperti pada gambar 23 berikut ini :

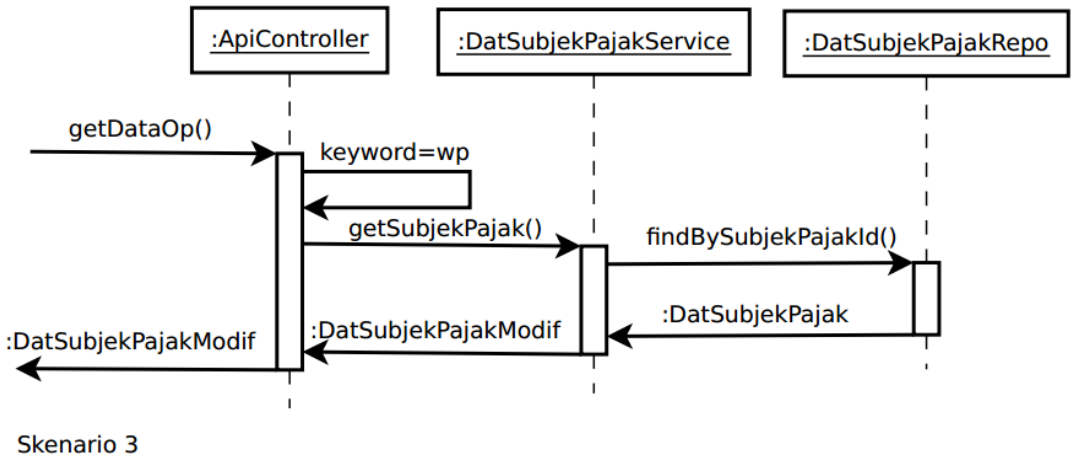

Figure 23: Diagram *Sequence* Untuk Skenario Ketiga

Untuk skenario keempat, sistem akan memberikan informasi tanggal jatuh tempo dan denda (apabila ada karena keterlambatan pembayaran) untuk objek pajak berdasarkan nomor objek pajak dan tahun pajak tertentu. Diagram *sequence* untuk skenario ini adalah seperti pada gambar 24 berikut ini :

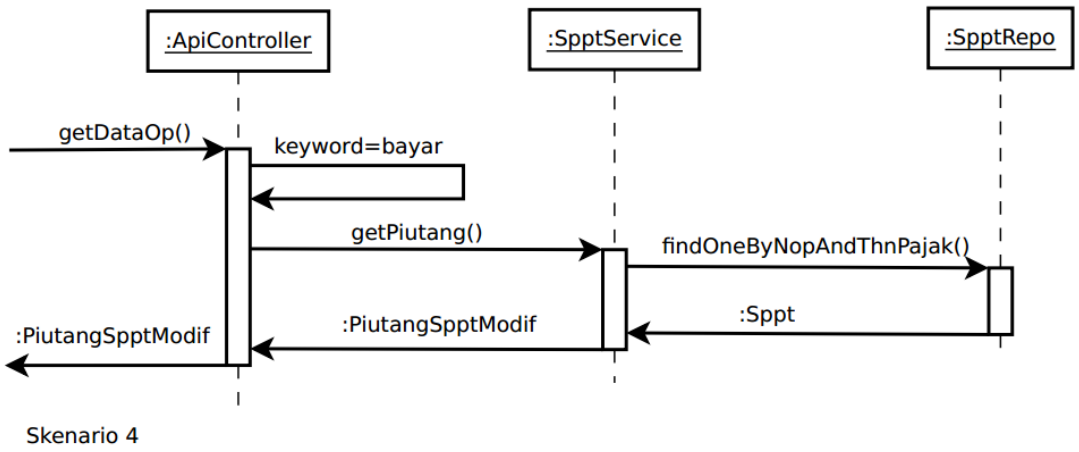

Figure 24: Diagram *Sequence* Untuk Skenario Keempat

5.3 Bagian Tampilan Aplikasi

Desain tampilan aplikasi yang diharapkan nantinya adalah seperti pada gambar 5.3 dan 5.3 berikut ini :

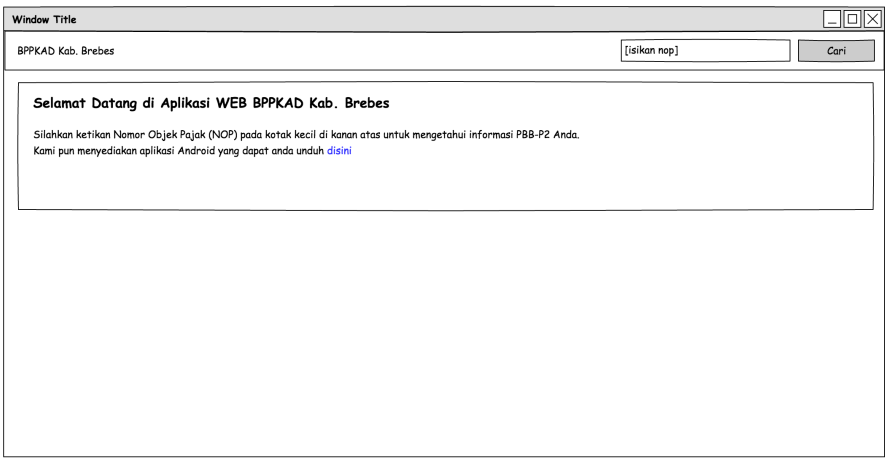

Figure 25: Tampilan Awal Aplikasi

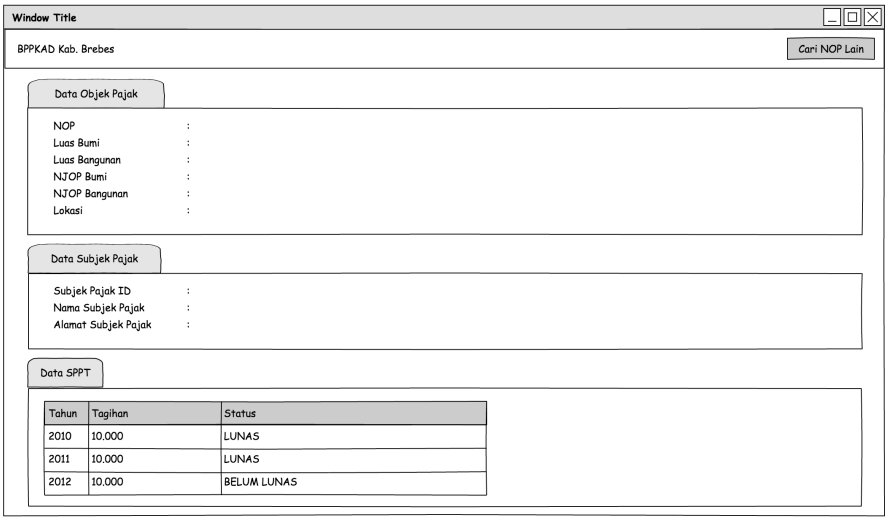

Figure 26: Tampilan Aplikasi Setelah Memperoleh Hasil

6 BIAYA DAN JADWAL PENGEMBANGAN

1. Biaya Pengembangan

   Pengembangan aplikasi ini memerlukan beberapa perangkat dan kelengkapannya seperti berikut ini :

   (a) Peladen Sistem Basis Data

   (b) Peladen Aplikasi *Web*

   (c) Aplikasi Peladen *Servlet*, dalam hal ini menggunakan Apache Tomcat yang tersedia gratis.

   (d) *Driver* JDBC yang tersedia gratis.

   (e) Akses Internet

   (f) IDE, menggunakan Intellij IDEA versi *Community Edition*

   Melihat ketersediaan perangkat keras dan perangkat lunak tersebut di atas sudah ada, hanya tinggal digunakan dan beberapa perangkat lunak hanya tinggal di unduh, maka tidak ada biaya yang diperlukan untuk pengembangannya.

2. Jadwal Pengembangan

   Jadwal pengembangan untuk membangun aplikasi atau sistem informasi ini tertuang pada diagram *gantt* seperti pada gambar **??** berikut ini :

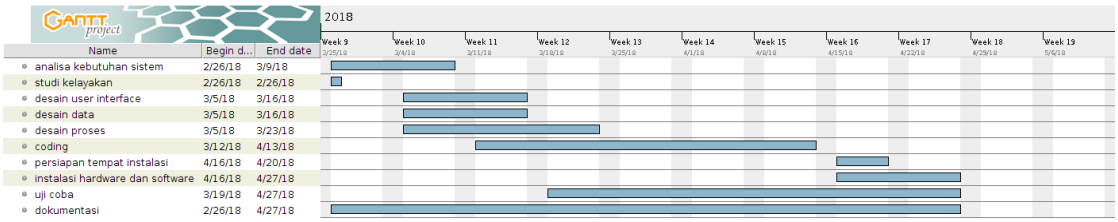

Figure 27: Diagram *Gantt* Untuk Jadwal Pengembangan Aplikasi