RANCANGAN RINCI SISTEM INFORMASI PEMBAYARAN PAJAK BUMI DAN BANGUNAN PERDESAAN DAN PERKOTAAN DI KABUPATEN BREBES

PERIODE PENILAIAN TAHUN 2018

Oleh :

Priyanto Tamami, S.Kom.

NIP 19840409 201001 1 025

Fungsional Pranata Komputer

Dinas Pendapatan dan Pengelolaan Keuangan

Pemerintah Daerah Kabupaten Brebes

Brebes, 12 Maret 2018

Lembar Pengesahan

Nama Kegiatan	:	Membuat Rancangan Sistem Informasi
Judul	:	RANCANGAN RINCI SISTEM INFORMASI PEMBAYARAN PAJAK BUMI DAN BANGUNAN PERDESAAN DAN PERKOTAAN DI KABUPATEN BREBES

Disetujui oleh :	Disusun Oleh
Kepala Sub Bidang Keberatan	Pranata Komputer
Pada tanggal 13 Maret 2018	Selesai tanggal : 12 Maret 2018

M.L. Setiyawan, S.E.Ak	Priyanto Tamami, S.Kom
NIP 19790530 200604 1 006	NIP 19840409 201001 1 025

Daftar Isi

Daftar Gambar

Bab 1

SISTEM KOMPUTER

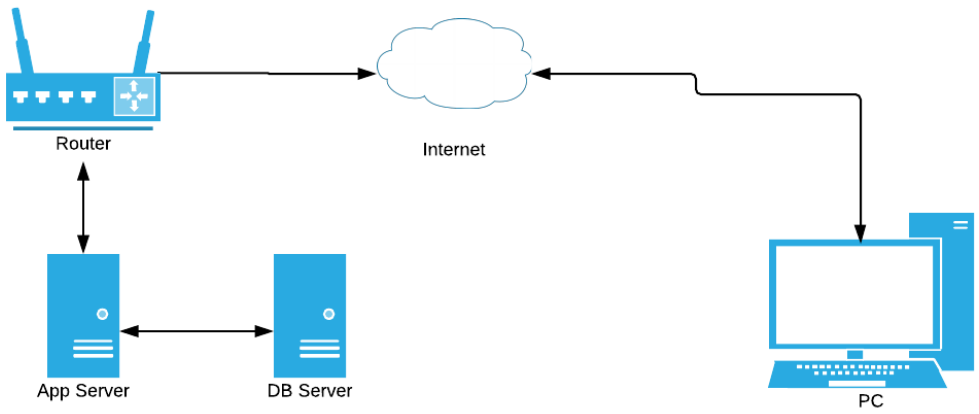
Sistem komputer yang digunakan akan terbagi menjadi 3 (tiga) bagian, yaitu :

1. Peladen sistem basis data dengan spesifikasi sebagai berikut :
    - Prosesor Intel Xeon 2,4 GHz.
    - Memori 44 GB
    - Sistem Operasi Windows Server 2008 R2 64 bit
2. Peladen aplikasi dengan spesifikasi sebagai berikut :
    - Prosesor Intel Xeon 2,4 GHz.
    - Memori 8 GB
    - Sistem Operasi Ubuntu Server 16.04
3. Klien dengan spesifikasi dapat melakukan akses jaringan atau internet dan telah terpasang peramban (*browser*) seperti Internet Explorer, Mozilla Firefox, Google Chrome, atau Safari. Karena nantinya yang akan melakukan akses dari sistem ini adalah masyarakat wajib pajak dengan menggunakan personal komputer dari manapun.

Bab 2

SISTEM JARINGAN

Sistem jaringan yang nantinya dibangun akan terlihat seperti pada gambar 2.1 berikut ini :



Gambar 2.1: Diagram Sistem Jaringan Sistem Informasi PBB-P2

Diagram tersebut menggambarkan bahwa peladen aplikasi akan berkomunikasi atau berhubungan langsung dengan peladen sistem basis data apabila ada permintaan data yang terjadi.

Untuk menghubungkan peladen aplikasi dengan dunia luar (internet) agar dapat diakses oleh masyarakat wajib pajak, maka diperlukan *router* yang akan menghubungkan jaringan lokal dengan jaringan internet.

Dari jaringan internet inilah masyarakat wajib pajak dapat melakukan akses ke aplikasi dan memperoleh informasi pembayaran tahun berapa saja yang menjadi hutang yang belum terbayar, dan pada tahun berapa saja yang sudah dilunasi.

Masyarakat wajib pajak pun dapat melihat informasi terkini mengenai data subjek pajak dan data objek pajak terbaru.

Bab 3

SISTEM BASIS DATA

Karena sistem informasi ini akan menampilkan status pembayaran dari pajak bumi dan bangunan sektor perdesaan dan perkotaan, maka sistem basis data yang digunakan adalah sama dengan sistem basis data yang digunakan pada SISMIOP (Sistem Manajemen Informasi Objek Pajak), karena sistem ini sebetulnya telah mengakomodir pencatatan pembayaran yang berbasis aplikasi *desktop*, aplikasi ini hanya melakukan akses terhadap data pencatatan pembayaran dengan berbasiskan *web* sehingga memungkinkan dapat diakses oleh masyarakat wajib pajak.

Aplikasi SISMIOP menggunakan sistem basis data Oracle 11g, namun tidak seluruh tabel digunakan untuk menampilkan informasi mengenai status pembayaran pajak bumi dan bangunan sektor perdesaan dan perkotaan, hanya beberapa tabel saja yang digunakan seperti pada daftar berikut :

- Tabel SPPT

  Tabel SPPT ini nantinya digunakan untuk menampilkan tahun pajak serta besar tagihan pajak terhutang untuk pajak bumi dan bangunan sektor perdesaan dan perkotaan, dan menampilkan status pembayaran untuk tiap tahun pajaknya, apabila muncul tagihan yang belum terbayarkan setelah melewati masa jatuh tempo, maka atas dasar besarnya pokok ketetapan pada tabel ini

pula akan dihitungkan jumlah sanksi administrasinya berupa denda setiap bulan sebesar 2% (dua persen) untuk maksimal 15 (lima belas) bulan.

- Tabel DAT_OBJEK_PAJAK

  Tabel DAT_OBJEK_PAJAK ini diakses untuk menampilkan informasi objek pajak terbaru yang tercetak dalam lembar Surat Pemberitahuan Pajak Terhutang yang diterbitkan tiap tahun.

  Data ini ditampilkan sebagai bahan verifikasi data apabila pernah melakukan perubahan data, sehingga subjek pajak atau wajib pajak dapat melakukan konfirmasi ke bagian pelayanan pajak daerah.

- Tabel DAT_SUBJEK_PAJAK

  Tabel DAT_SUBJEK_PAJAK ini diakses untuk menampilkan informasi subjek pajak terbaru yang dapat ditetapkan sebagai wajib pajak pada lembar Surat Pemberitahuan Pajak Terhutang (SPPT) untuk jenis pajak bumi dan bangunan sektor perdesaan dan perkotaan.

  Data ini pun ditampilkan sebagai bahan verifikasi data apabila pernah melakukan perubahan data yang biasanya akibat mutasi objek pajak, sehingga subjek pajak atau wajib pajak dapat melakukan konfirmasi ke bagian pelayanan pajak daerah.

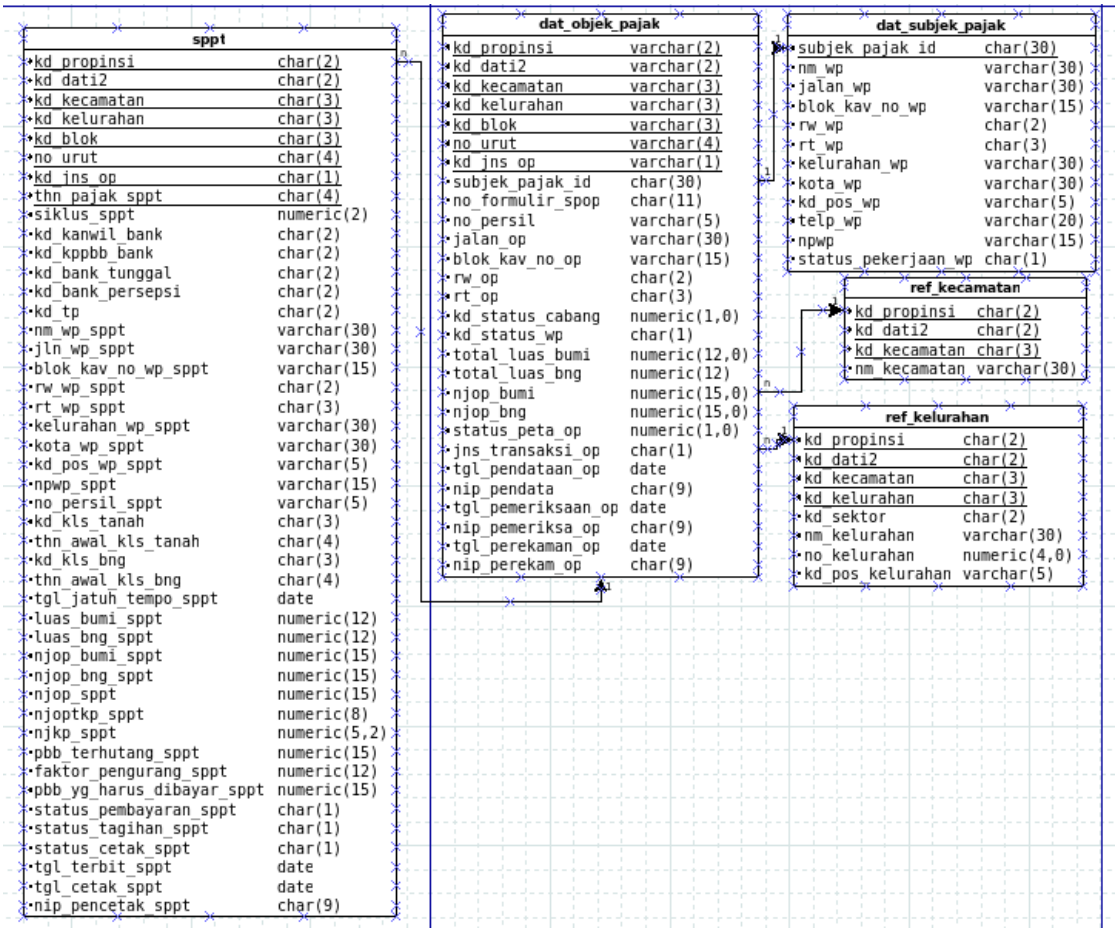
- Tabel REF_KECAMATAN

  Tabel REF_KECAMATAN diakses untuk menampilkan nama Kecamatan dari objek pajak yang diakses oleh pengguna.

- Tabel REF_KELURAHAN

  Tabel REF_KELURAHAN diakses untuk menampilkan nama Kelurahan dari objek pajak yang diakses oleh pengguna.

Diagram relasi tabel untuk tabel-tabel di atas adalah seperti pada gambar 3.1 berikut ini :



Gambar 3.1: Diagram Relasi Antar Tabel

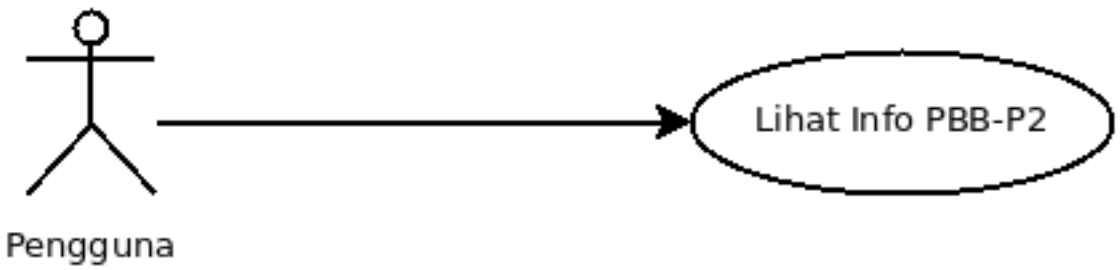
Bab 4

PROSEDUR AKTIVITAS

Prosedur aktivitas yang berlaku untuk sistem informasi pembayaran pajak bumi dan bangunan sektor perdesaan dan perkotaan ini dituangkan dalam beberapa diagram untuk mempermudah pemahaman alur aktivitas yang terjadi. Diagram tersebut adalah seperti berikut ini :

4.1 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 informasi ini seperti terlihat pada gambar 4.1 berikut ini :



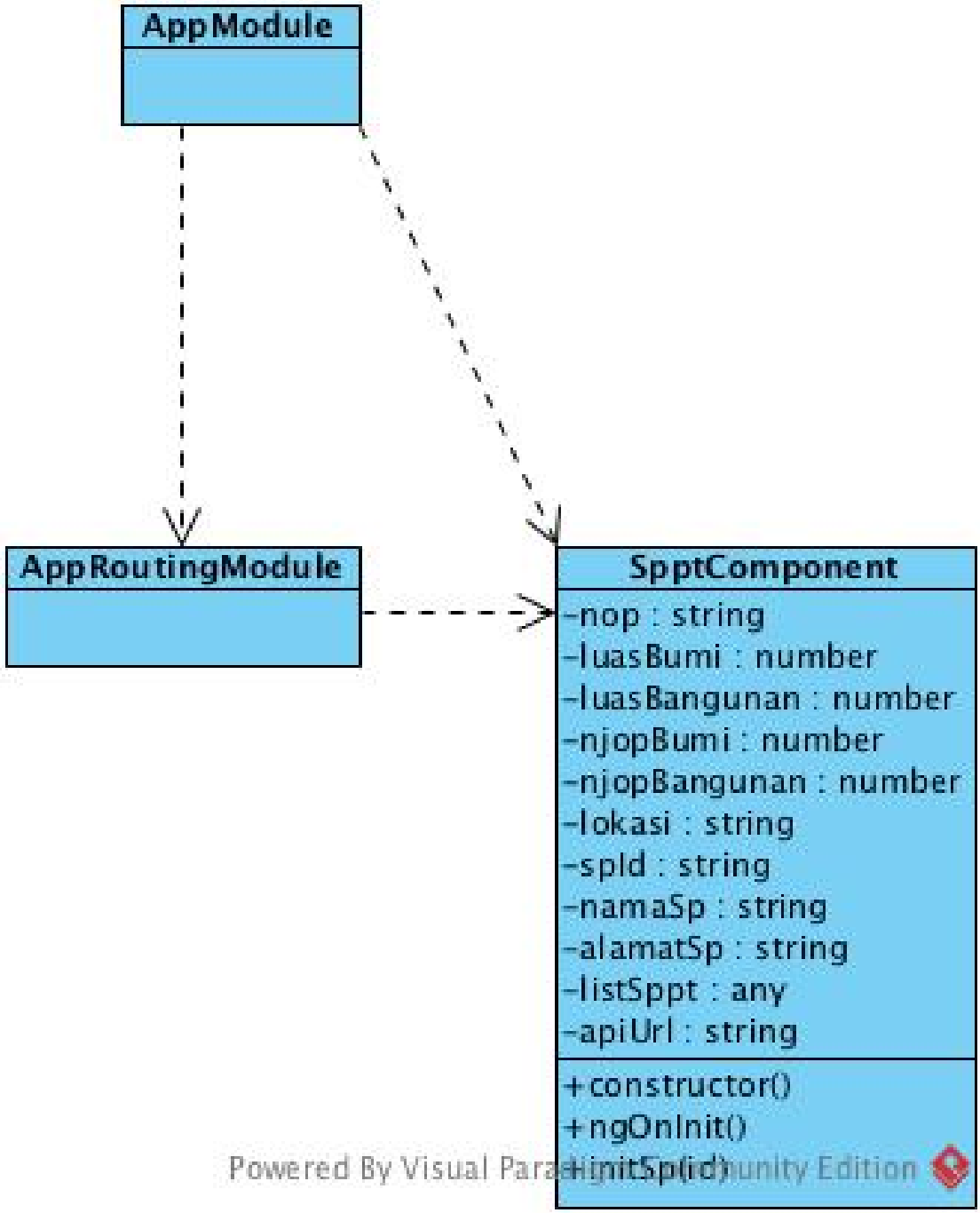
Gambar 4.1: Diagram *Use-Case* Sistem Informasi Pembayaran PBB-P2

Yang menjadi aktor disini adalah masyarakat wajib pajak yang melakukan akses ke sistem informasi pembayaran Pajak Bumi dan Bangunan Perdesaan dan Perkotaan (PBB-P2).

4.2 Diagram *Class*

Diagram *class* ini akan menggambarkan hubungan dari tiap kelas yang membangun sistem informasi ini menjadi utuh. Diagram ini akan dibagi menjadi 2 (dua) yang menggambarkan masing-masing bagian antara ujung-depan (*frontend*) dan ujung-belakang (*backend*).

Diagram *class* yang terdapat pada ujung depan (*frontend*) ini seperti pada gambar 4.2 berikut ini :



Gambar 4.2: Diagram *Class* Untuk Bagian Ujung Depan (*Frontend*)

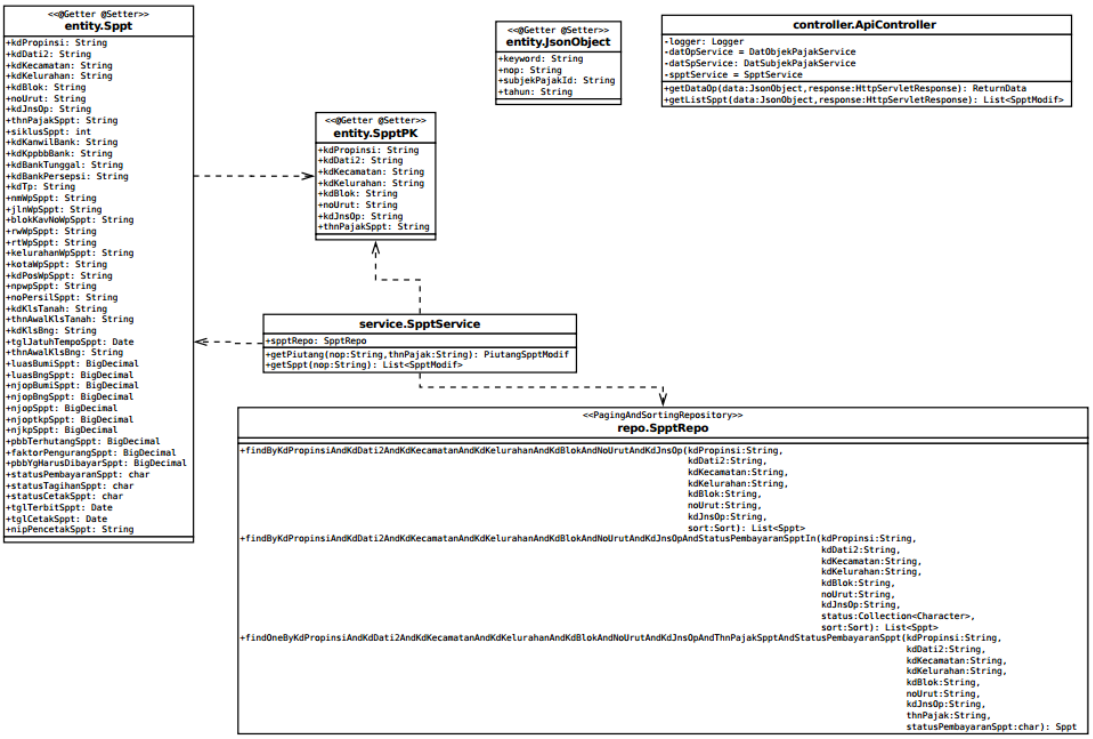
Pada bagian ujung depan (*frontend*), aplikasi akan berakar pada kelas `AppModule`, kelas `AppRoutingModule` yang akan mengatur bagaimana tiap komponen kelas (dalam hal ini halaman) akan berganti-ganti sesuai dengan kejadian yang diinginkan oleh pengguna.

Kelas `AppComponent` 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 `AppComponent` atau `SpptComponent`.

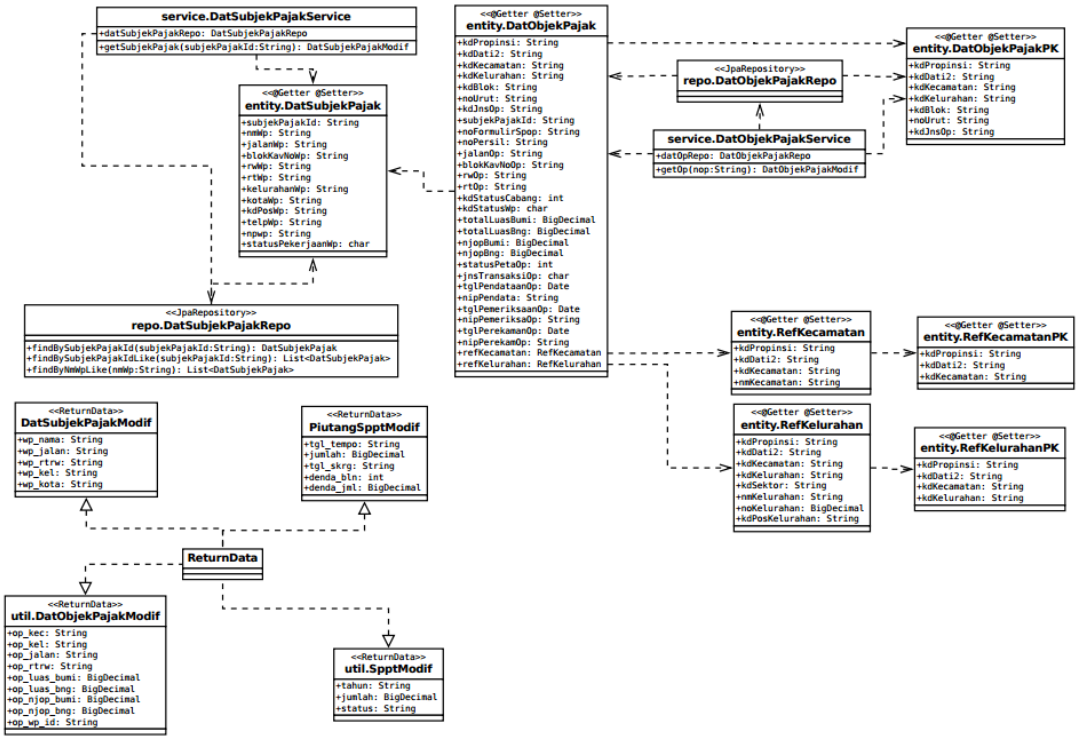
Kelas `AppComponent` sebetulnya kelas yang bertugas menjadi tampilan utama aplikasi, artinya begitu ada *browser* / peramban yang melakukan akses, halaman dari kelas `AppComponent` ini akan tampil terlebih dahulu menyambut pengguna.

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

Diagram `class` untuk bagian ujung belakang (*backend*) adalah seperti pada gambar 4.3 dan 4.4 berikut ini :



Gambar 4.3: Diagram *Class* Untuk Ujung Belakang (*backend*) Bagian 1



Gambar 4.4: Diagram *Class* Untuk Ujung Belakang (*backend*) 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 dirujuk 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 diikutsertakan dalam blok informasi ini sesuai dengan nama propertinya.
- Kelas `PiutangSpptModif`, kelas ini 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.

4.3 Diagram *Component*

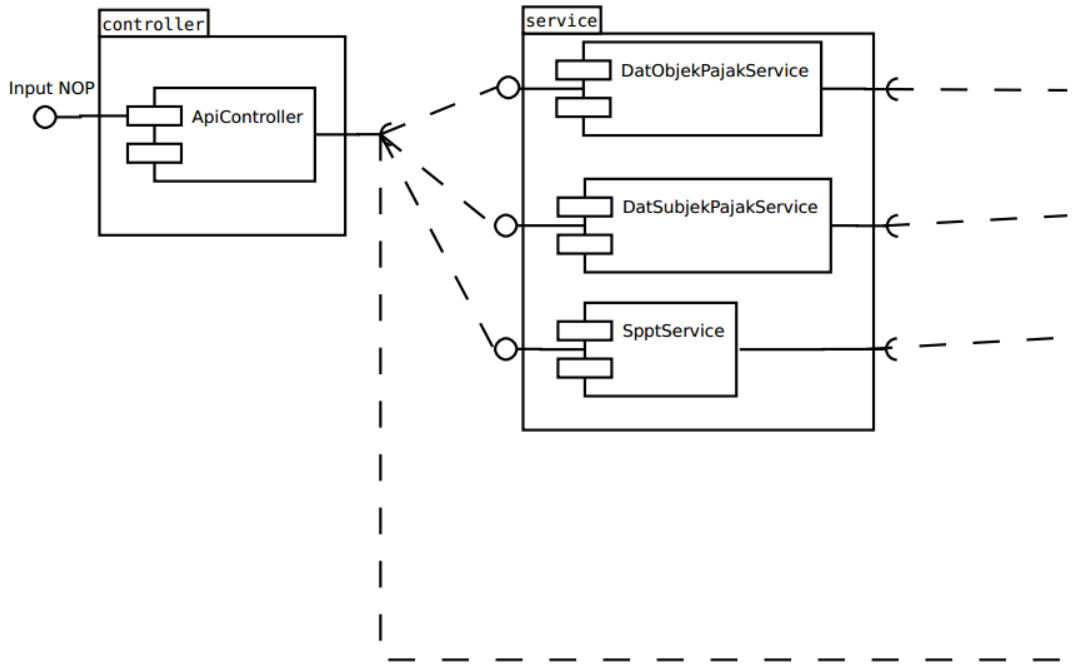
Diagram ini memberikan gambaran hubungan antar komponen, komponen mana yang membutuhkan data dan komponen mana yang memberikan data akan terlihat jelas pada diagram komponen ini.

Diagram *component* untuk bagian ujung-depan (*frontend*) ditunjukkan seperti pada gambar 4.3 berikut ini :

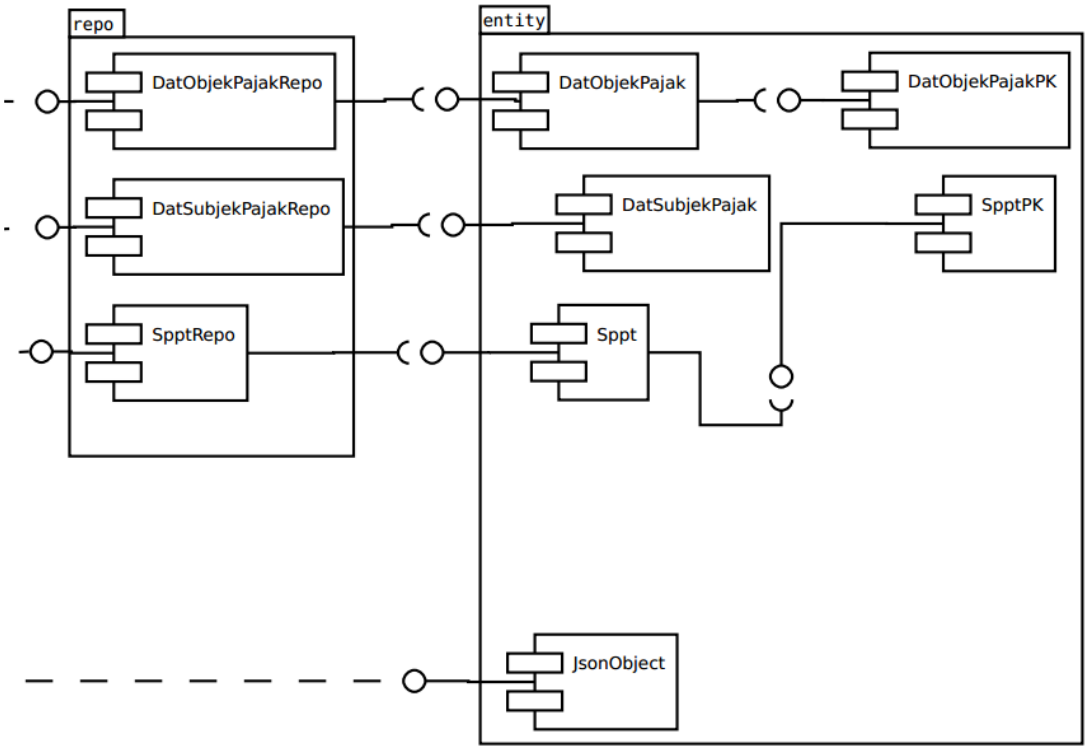
Gambar 4.5: Diagram *Component* Untuk Ujung-Depan (*frontend*)

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.

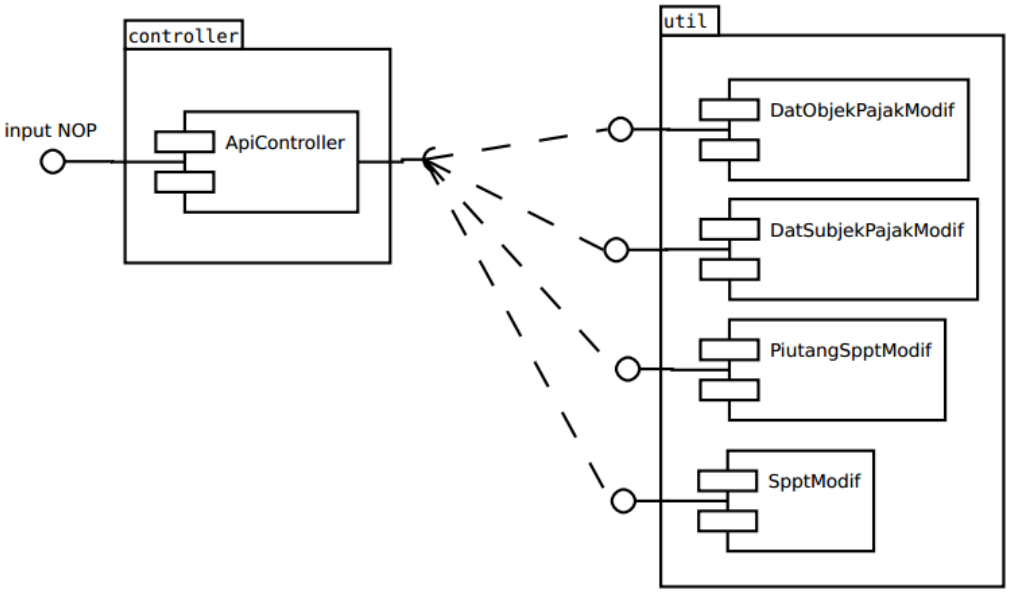
Diagram *component* yang membentuk bagian ujung belakang (*backend*) adalah seperti pada gambar 4.6, 4.7, dan 4.8 berikut ini :



Gambar 4.6: Diagram *Component* Bagian 1



Gambar 4.7: Diagram *Component* Bagian 2



Gambar 4.8: 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 7.

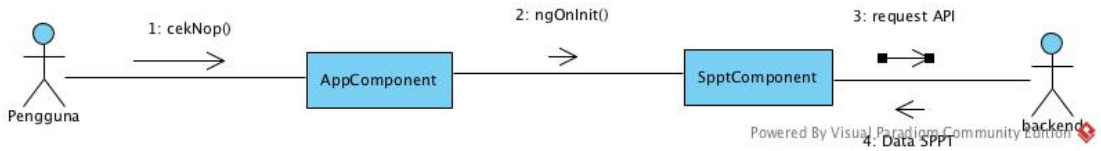
Kemudian tiap *service* akan melakukan pemanggilan data terhadap *repository*-nya masing-masing yang kemudian akan dikembalikan dalam bentuk objek sesuai tabel yang diaksesnya seperti pada gambar 4.6 dan 4.7.

Hasil yang dikembalikan adalah beberapa objek dari kelas pada paket `util` seperti disebutkan pada gambar 4.8.

4.4 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 4.4 berikut ini :

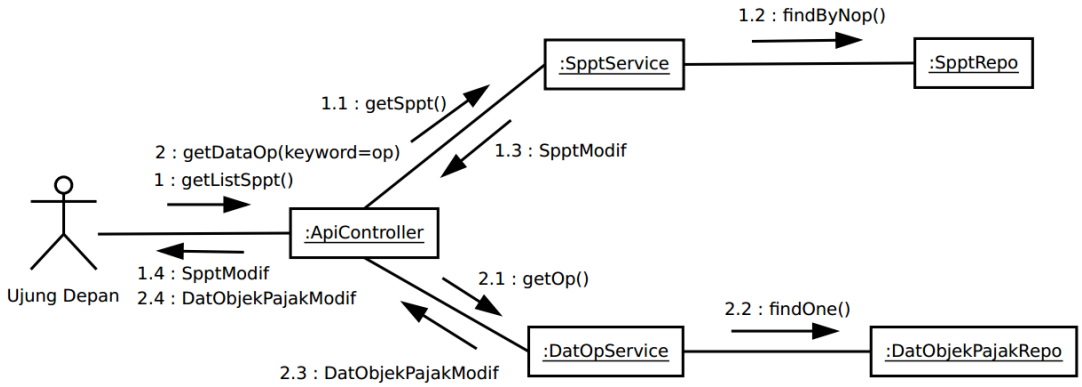


Gambar 4.9: Diagram *Communication* Untuk Bagian Ujung-Depan (*frontend*)

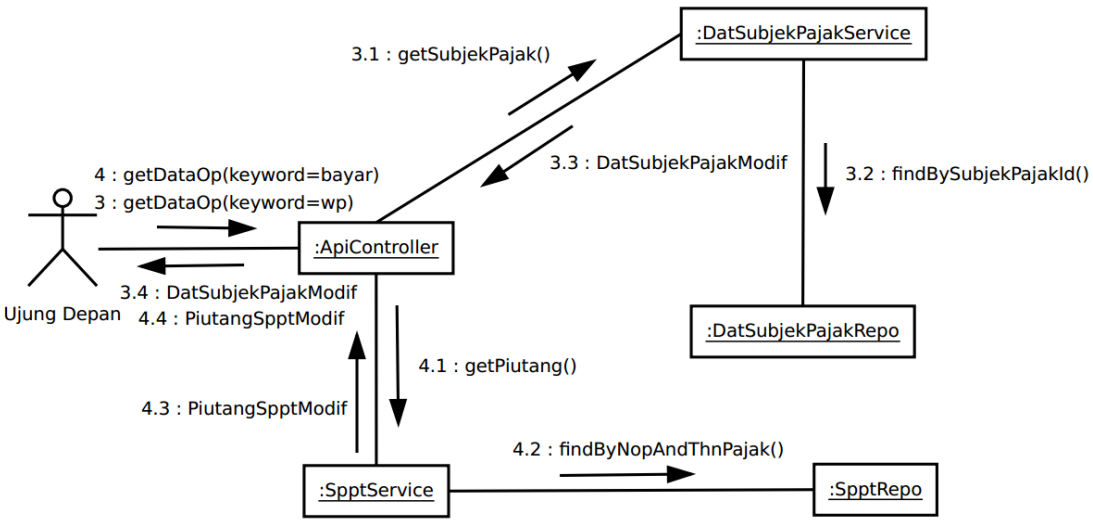
Alurnya adalah dari pengguna akan memasukkan Nomor Objek Pajak (NOP) pada kolom yang disediakan pada `AppComponent`, kemudian komponen ini akan memanggil fungsi `cekNop()` dan mengaktifkan `SpptComponent` yang secara otomatis akan memanggil fungsi `ngOnInit()`.

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 4.10 dan 4.11 berikut ini :



Gambar 4.10: Diagram *Communication* Bagian 1



Gambar 4.11: Diagram *Communication* Bagian 2

Nomor urut 1 (satu) dan 2 (dua) pada saat ujung-depan (*frontend*) 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 alah urutan komunikasi dari skenario 1 (satu) dari langkah pertama

sampai langkah ke-n.

Jadi untuk skenario yang ke-1 (satu), bagian ujung-depan (*frontend*) 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 (*frontend*) berupa objek dari kelas `SpptModif` (pada langkah 1.3 dan 1.4).

Pada skenario ke-2, ujung-depan (*frontend*) 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 pada bagian ujung-depan (*frontend*) dalam bentuk JSON.

Pada skenario ke-3 di gambar 4.11, aplikasi bagian ujung-depan (*frontend*) akan melakukan *request* ke *method* `getDataOp()` dengan parameter `keyword` berisi teks `wp`. Langkah selanjutnya adalah memanggil *method* `getSubjekPajak()` milik kelas `DatSubjekPajakService`, yang prosesnya kemudian akan memanggil *method* `findBySubjekPajakId()`, hasilnya kemudian akan disesuaikan masuk ke dalam kelas `DatSubjekPajakModif` yang dikembalikan ke bagian ujung-depan (*frontend*).

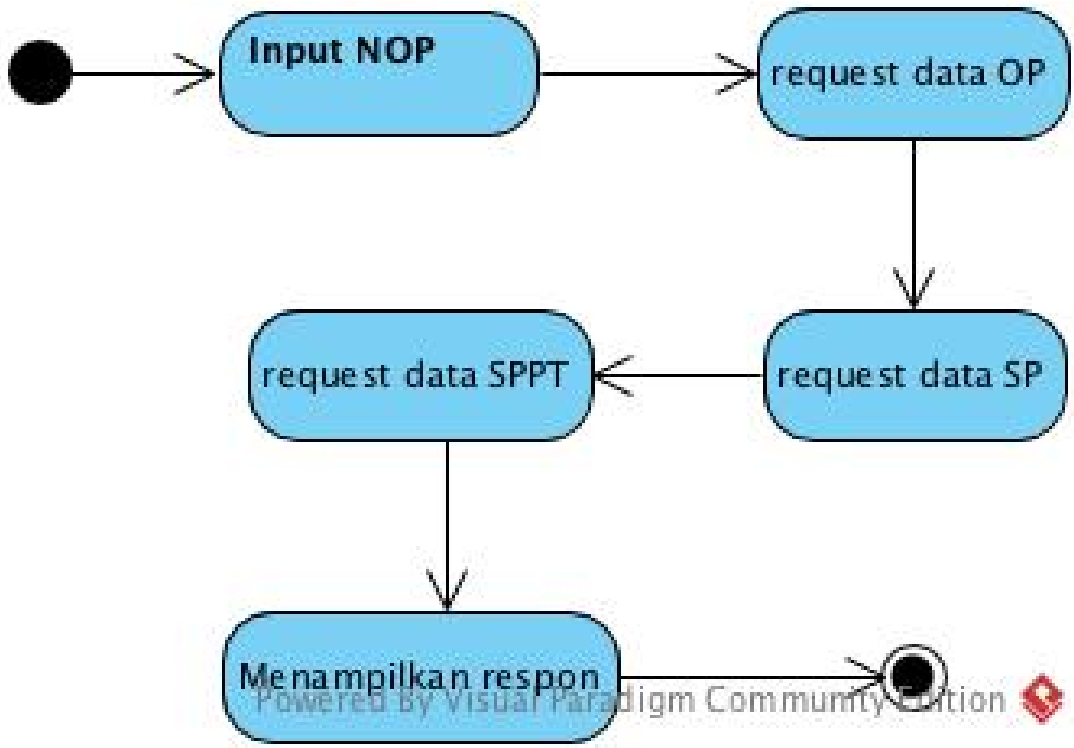
Skenario ke-4 sama saja seperti skenario ke-3 dan ke-2, akan memanggil *method* `getDataOp()` dengan parameter *keyword* berisi teks `bayar`. Langkah selanjutnya adalah memanggil *method* `getPiutang()` milik kelas `SpptService` yang kemudian akan mengambil data ke sistem basis data dengan memanggil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

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

4.5 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 4.12 berikut ini :



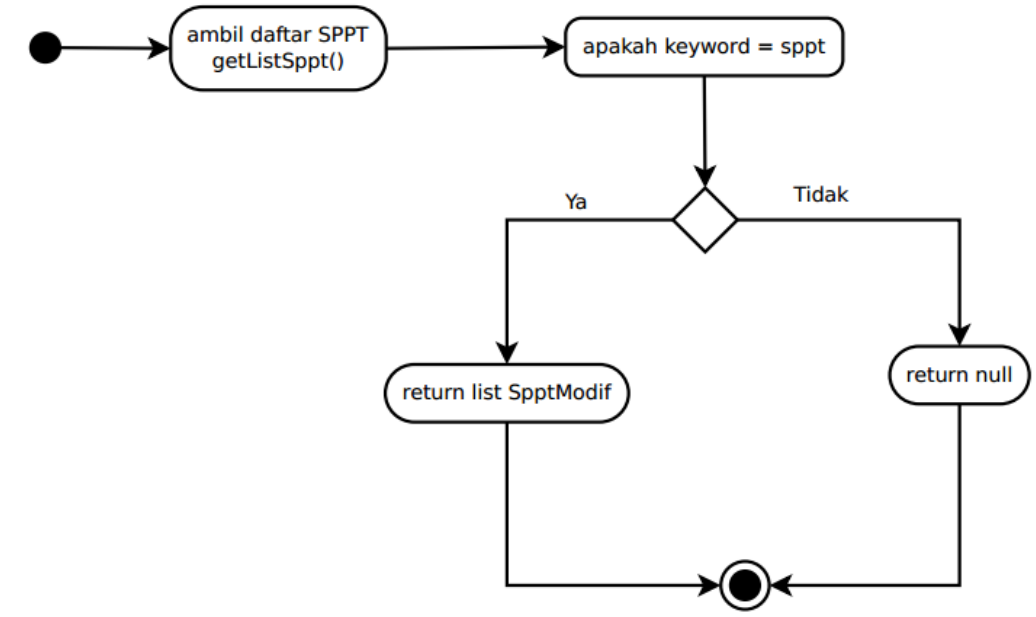
Gambar 4.12: Diagram *Activity* Untuk Bagian Ujung-Depan (*frontend*)

Terlihat pada diagram tersebut bahwa bagian ujung-depan (*frontend*) akan melakukan 3 (tiga) kali *request* atau permintaan data ke peladen API, kemudian

menampilkan hasil yang diberikan oleh peladen ke jendela *browser* / peramban.

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

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 4.13 berikut ini :

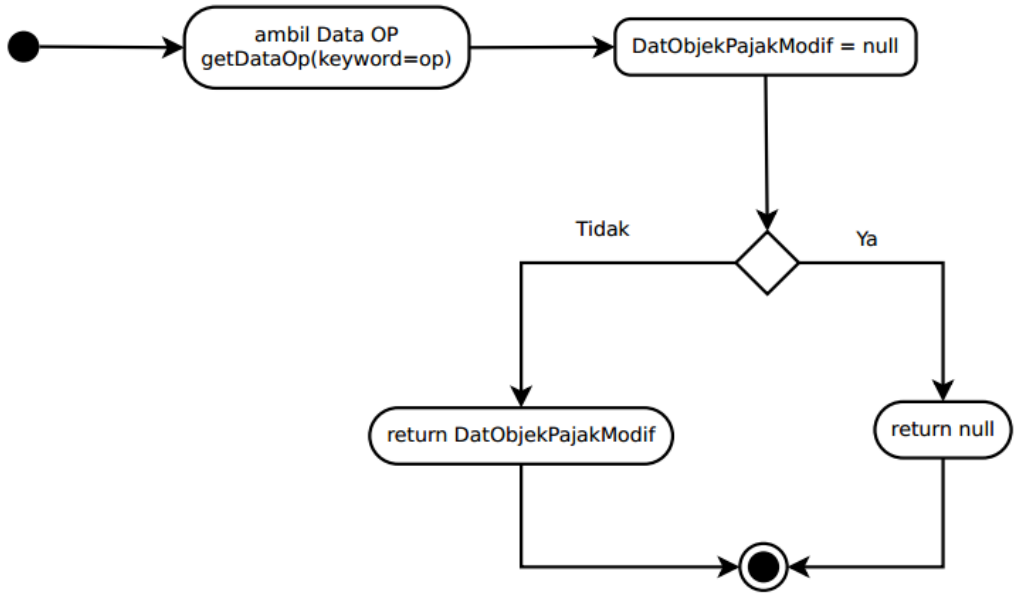


Gambar 4.13: Diagram *Activity* Untuk Ambil Daftar Tagihan

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

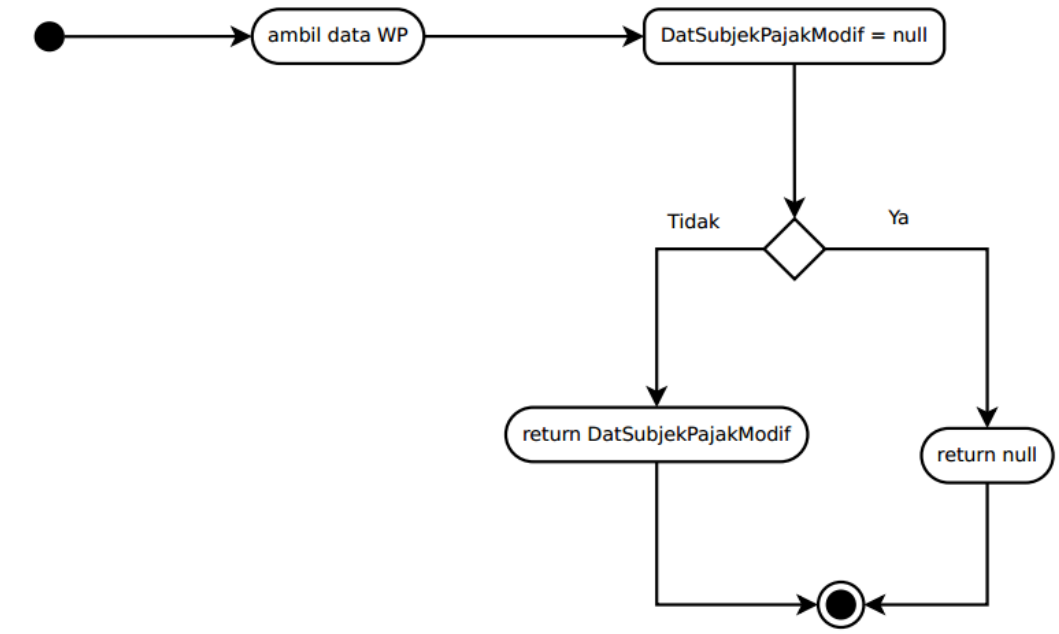
Skenario yang kedua terjadi ketika ada *request* data objek pajak dari klien. Diagram *activity* untuk skenario kedua ini seperti terlihat pada gambar 4.14 berikut

ini :



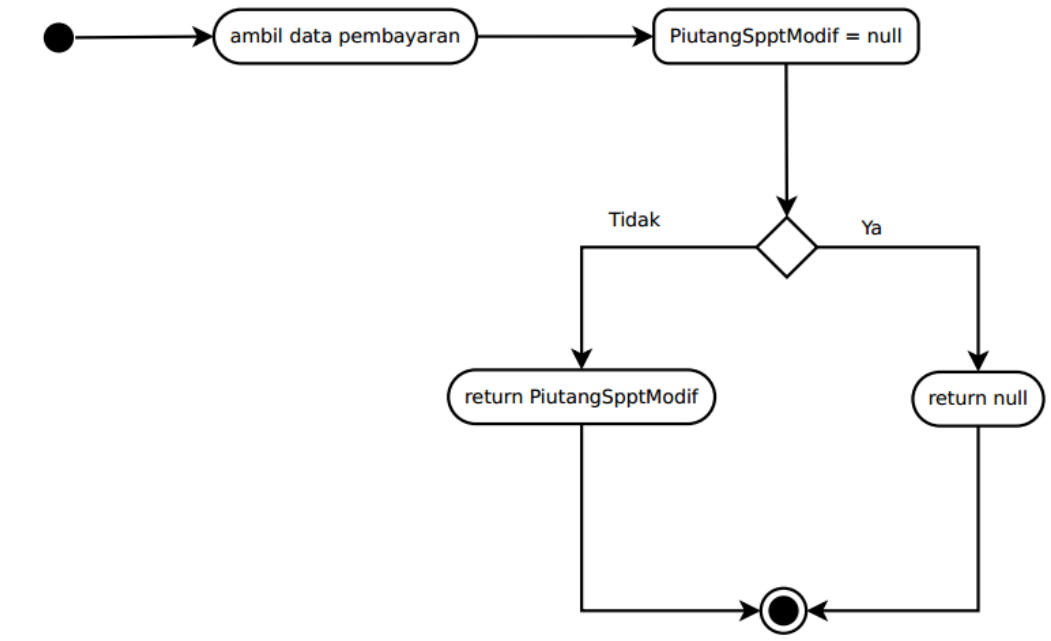
Gambar 4.14: Diagram *Activity* Skenario Kedua

Skenario berikutnya adalah skenario ketiga, yang terjadi ketika ada *request* data wajib pajak dari klien. Diagram *activity* untuk skenario ini adalah seperti pada gambar 4.15 berikut ini :



Gambar 4.15: Diagram *Activity* Skenario Ketiga

Skenario keempat terjadi ketika ada *request* data pembayaran dari klien. Diagram *activity* untuk skenario ini adalah seperti pada gambar 4.16 berikut ini :

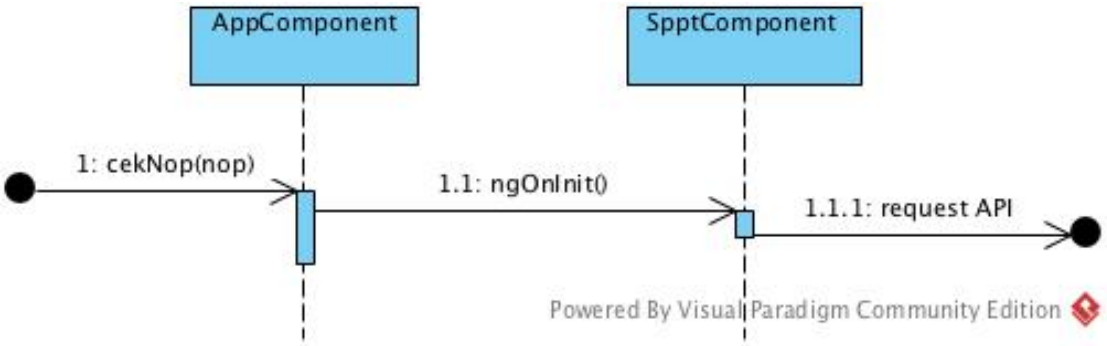


Gambar 4.16: Diagram *Activity* Untuk Skenario Keempat

4.6 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 4.17 berikut ini :



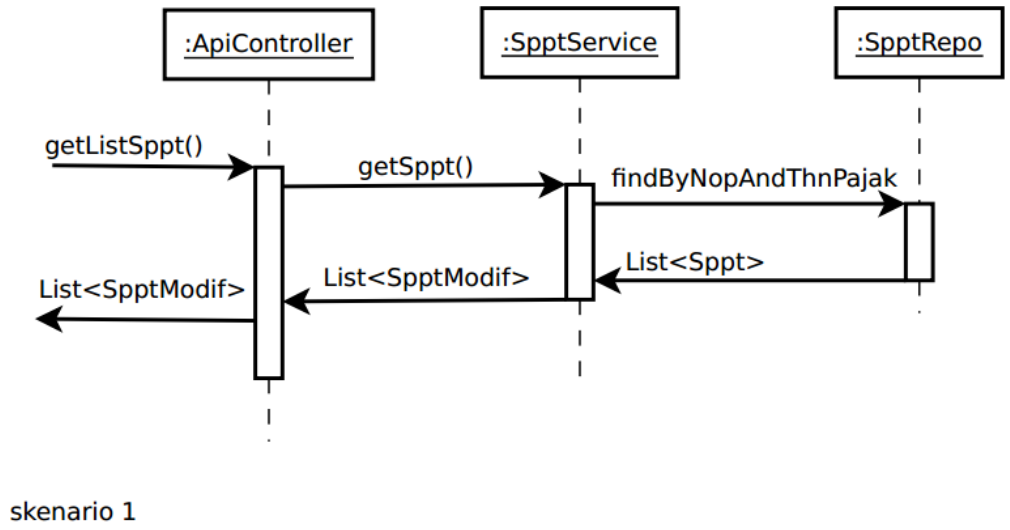
Gambar 4.17: Diagram *Sequence* Untuk Bagian Ujung-Depan (*frontend*)

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.

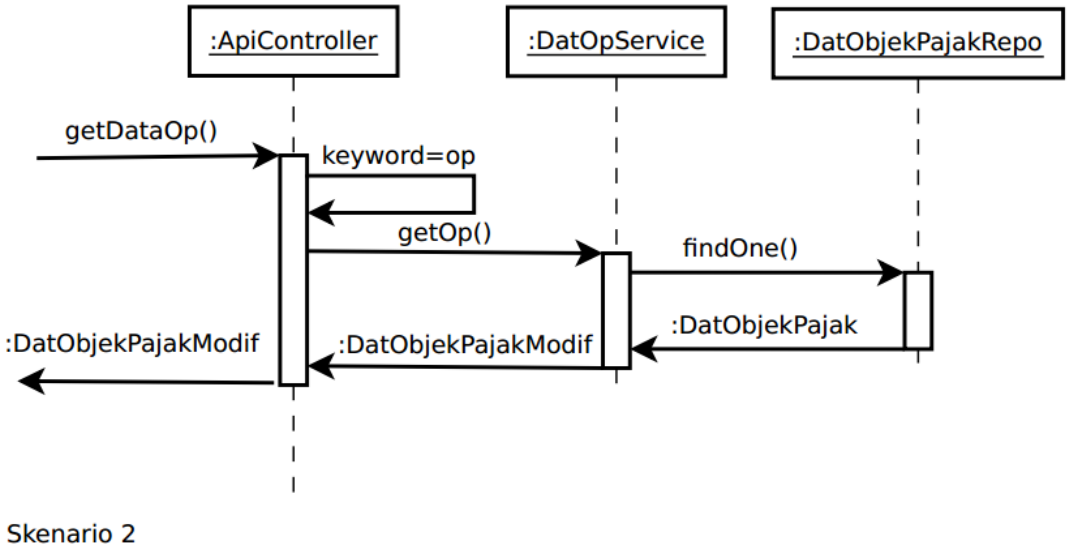
Pada skenario pertama, sistem akan memberikan daftar tagihan pajak bumi dan bangunan sektor perdesaan dan perkotaan untuk seluruh wajib pajak berdasarkan nomor objek pajak yang diminta.

Diagram *sequence* untuk skenario pertama ini seperti pada gambar 4.18 berikut ini :



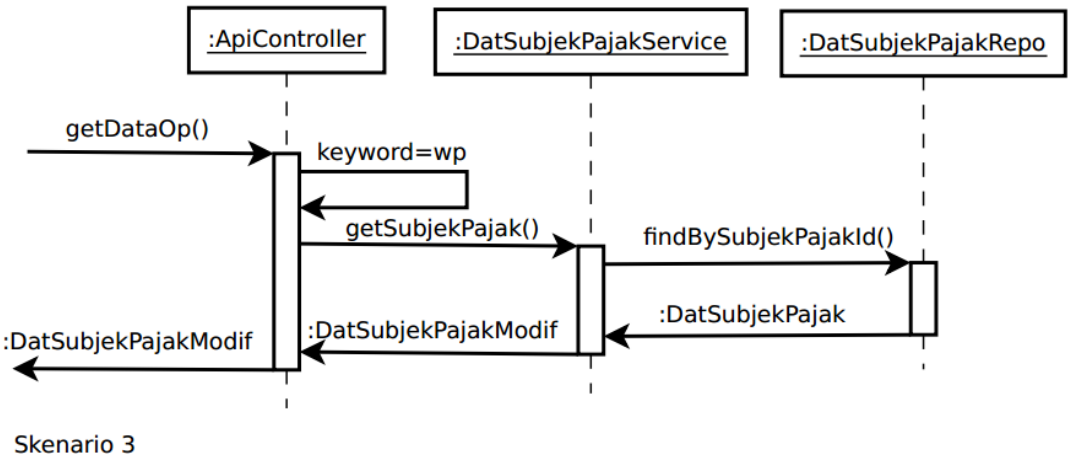
Gambar 4.18: Diagram *Sequence* Untuk Skenario Pertama

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 **??** berikut ini :



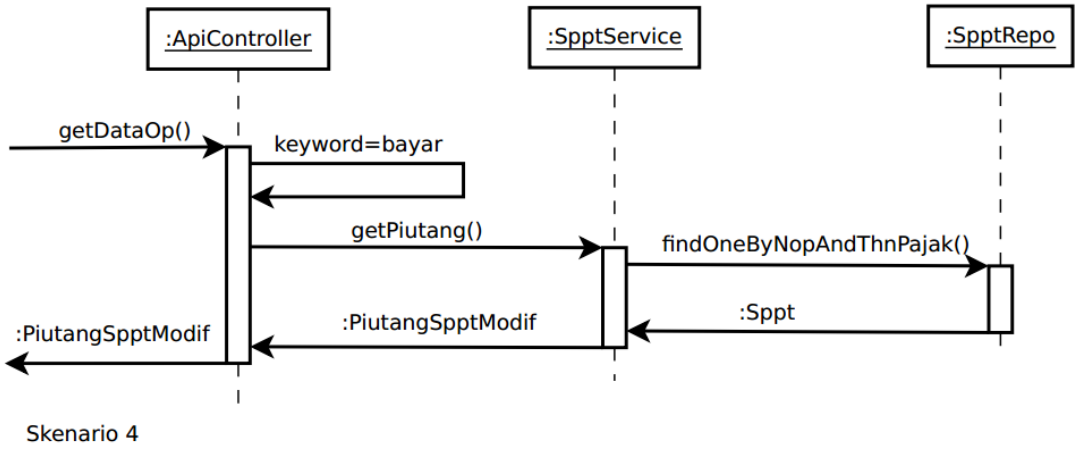
Gambar 4.19: Diagram *Sequence* Untuk Skenario Kedua

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 4.20 berikut ini :



Gambar 4.20: 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 4.21 berikut ini :

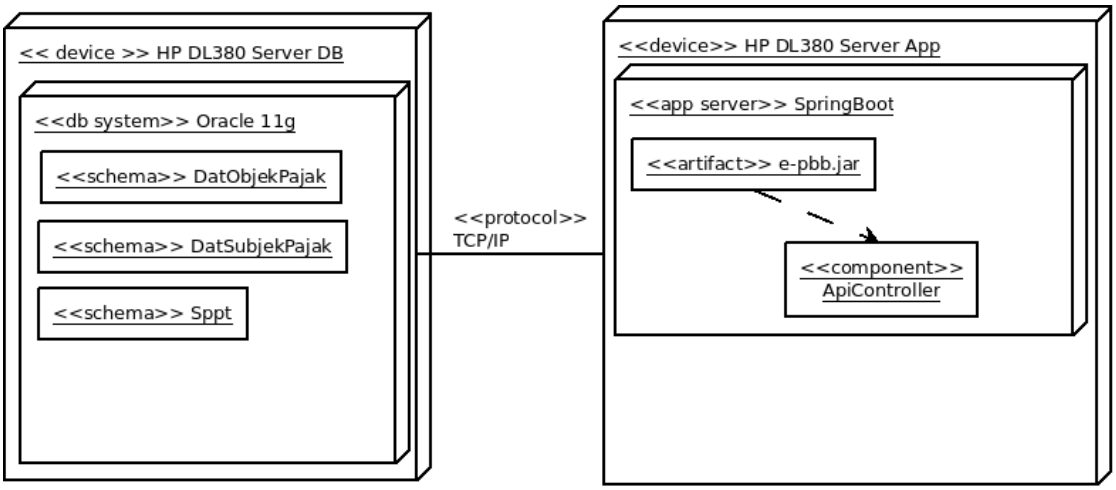


Gambar 4.21: Diagram *Sequence* Untuk Skenario Keempat

4.7 Diagram *Deployment*

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

Diagram ini digambarkan seperti pada gambar 4.22 berikut ini :



Gambar 4.22: 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 sebagai aplikasi dengan Tomcat dan lingkungan berada dalam satu paket pada Springboot.

Bab 5

SUMBER DAYA MANUSIA

Karena aplikasi ini ditujukan bagi masyarakat wajib pajak, maka tidak perlu ada petugas tambahan untuk mengoperasikan sistem informasi ini. Aplikasi diharapkan mampu atau dapat digunakan oleh pengguna komputer awam sekalipun.