ALGORITMA PEMROGRAMAN UNTUK 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

Badan Pengelolaan Pendapatan, Keuangan dan Aset Daerah

Pemerintah Daerah Kabupaten Brebes

Brebes, 28 Maret 2018

Lembar Pengesahan

Nama Kegiatan	:	Membuat Algoritma Pemrograman
Judul	:	ALGORITMA PEMROGRAMAN UNTUK 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 29 Maret 2018	Selesai tanggal : 28 Maret 2018

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

Daftar Isi

1 PENGANTAR **1**

2 DIAGRAM UNIFIED MODELING LANGUAGE (UML) **2**

2.1 Diagram *Use-Case* 2

2.2 Diagram *Class* 3

2.3 Diagram *Component* 8

2.4 Diagram *Communiation* 11

2.5 Diagram *Activity* 14

2.6 Diagram *Sequence* 19

2.7 Diagram *Deployment* 23

Daftar Gambar

2.1 Diagram *Use-Case* Sistem Informasi Pembayaran PBB-P2 2
2.2 Diagram *Class* Untuk Bagian Ujung-Depan (*frontend*) 3
2.3 Diagram *Class* Untuk Ujung Belakang (*backend*) Bagian 1 4
2.4 Diagram *Class* Untuk Ujung Belakang (*backend*) Bagian 2 5
2.5 Diagram *Component* Untuk Ujung-Depan (*frontend*) 9
2.6 Diagram *Component* Bagian 1 9
2.7 Diagram *Component* Bagian 2 10
2.8 Diagram *Component* Bagian 3 10
2.9 Diagram *Communication* Untuk Bagian Ujung-Depan (*frontend*) . . 11
2.10 Diagram *Communication* Bagian 1 12
2.11 Diagram *Communication* Bagian 2 12
2.12 Diagram *Activity* Untuk Bagian Ujung-Depan (*frontend*) 15
2.13 Diagram *Activity* Untuk Ambil Daftar Tagihan 16
2.14 Diagram *Activity* Skenario Kedua 17
2.15 Diagram *Activity* Skenario Ketiga 18
2.16 Diagram *Activity* Skenario Keempat 19
2.17 Diagram *Sequence* Untuk Bagian Ujung-Depan (*frontend*) 20
2.18 Diagram *Sequence* Untuk Skenario Pertama 21
2.19 Diagram *Sequence* Untuk Skenario Kedua 21

2.20 Diagram *Sequence* Untuk Skenario Ketiga 22
2.21 Diagram *Sequence* Untuk Skenario Keempat 23
2.22 Diagram *Deployment* 24

Bab 1

PENGANTAR

Untuk membangun sistem ini akan menggunakan bahasa pemrograman Java dengan *framework* Springboot sebagai *service* atau peladen pada bagian belakang (*backend*). Sedangkan pada bagian depan (*frontend*) menggunakan Angular.

Karena basis pemrograman dari Java menggunakan orientasi objek, maka kurang tepat apabila digambarkan dalam diagram *flowchart* yang alur datanya dari atas ke bawah, karena karakteristik pemrograman berorientasi objek alur datanya akan berpindah pindah antar kelas dan *method*.

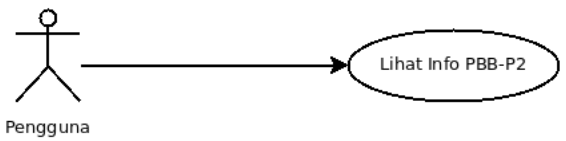
Ilustrasi yang tepat untuk menggambarkan logika aplikasi dari sistem yang akan dibangun adalah menggunakan UML (*Unified Modelling Language*), dimana diagram akan dipecah kedalam beberapa skenario atau *case* yang telah dibentuk pada saat pengambilan *requirement*.

Bab 2

DIAGRAM UNIFIED MODELING LANGUAGE (UML)

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 informasi ini seperti terlihat pada gambar 2.1 berikut ini :



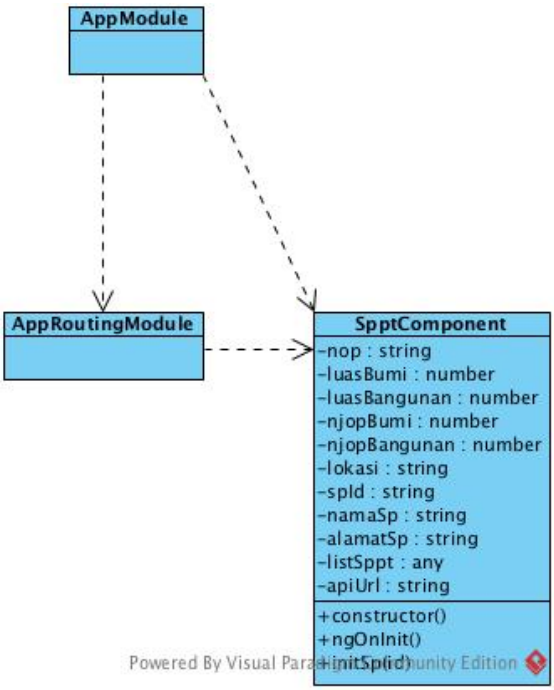
Gambar 2.1: Diagram *Use-Case* Sistem Informasi Pembayaran PBB-P2

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

2.2 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 *fig:class-dia-fe* berikut ini :



Gambar 2.2: 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.

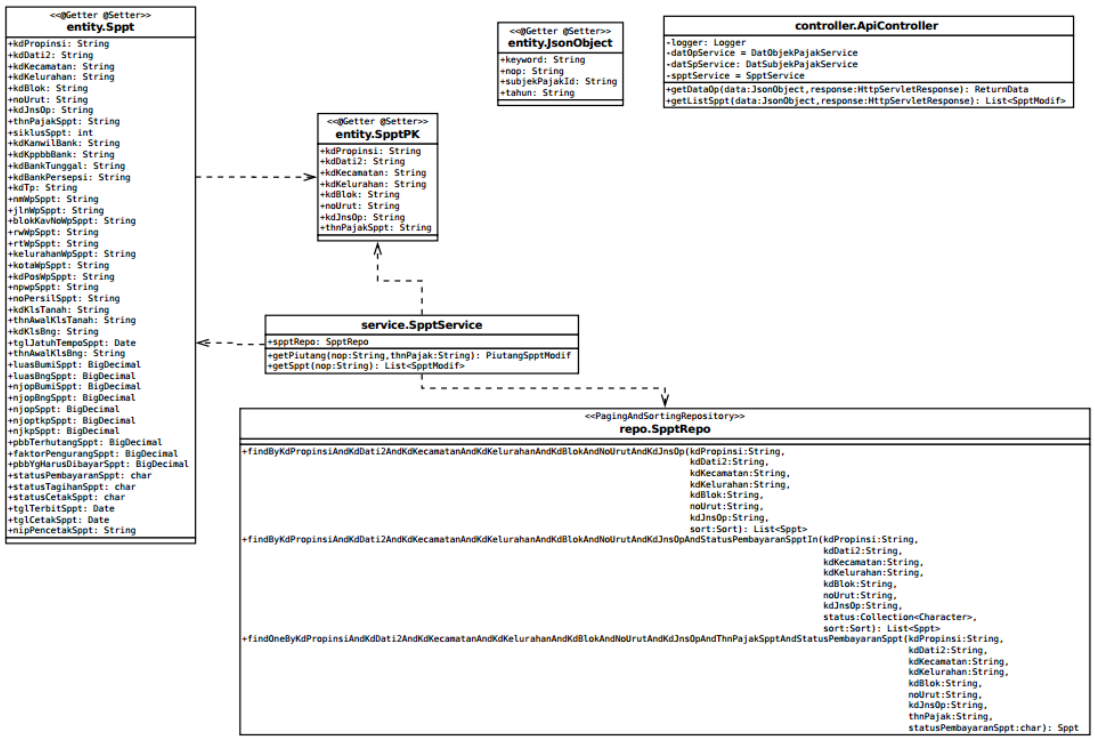
Kelas `AppComponent` dan `SpptComponent` adalah 2 (dua) kelas yang berhubung-

an langsung dengan tampilan aplikasi (*view*), keduanya akan dikontrol langsung oleh kelas `AppRoutingModule` bilamana halaman yang ditampilkan terlebih dahulu apakah `AppComponent` atau `SpptComponent`.

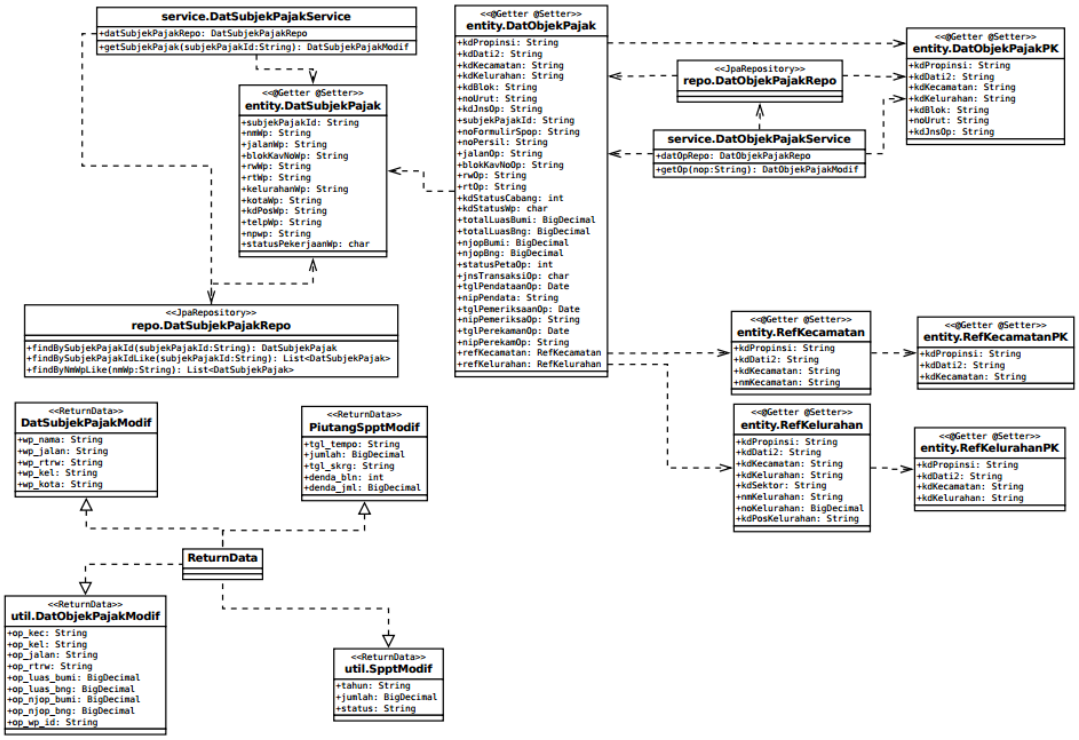
Kelas `AppComponent` sebetulnya kelas yang bertugas menjadi tampilan utama aplikasi, artinya begitu ada *browser* / peramban yang melakukan akses, halaman dari kelas `AppComponent` ini akan tampil terlebih dahulu menyambut pengguna.

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 2.3 dan 2.4 berikut ini :



Gambar 2.3: Diagram *Class* Untuk Ujung Belakang (*backend*) Bagian 1



Gambar 2.4: 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 (NOP) 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 (PBB-P2) sesuai dengan Nomor Objek Pajak (NOP) 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 (NOP) 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.

2.3 Diagram *Component*

Diagram ini memberikan gambaran hubungan antar komponen, komponen mana yang membutuhkan data dan komponen mana yang memberikan data akan terlihat

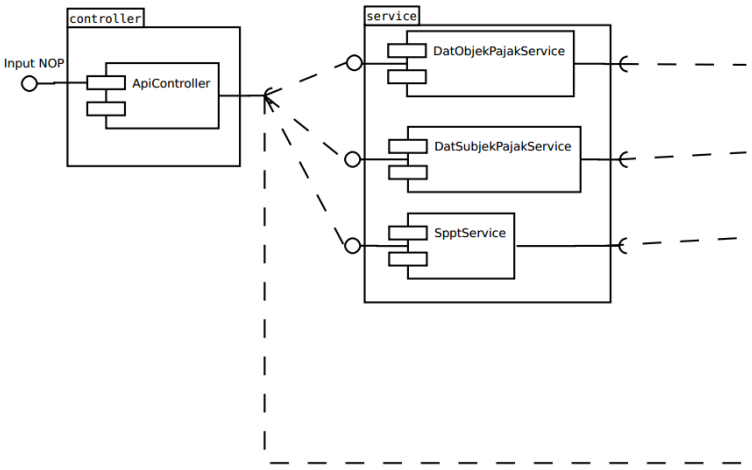
jelas pada diagram komponen ini.

Diagram *component* untuk bagian ujung-depan (*frontend*) ditunjukkan seperti pada gambar 2.5 berikut ini :

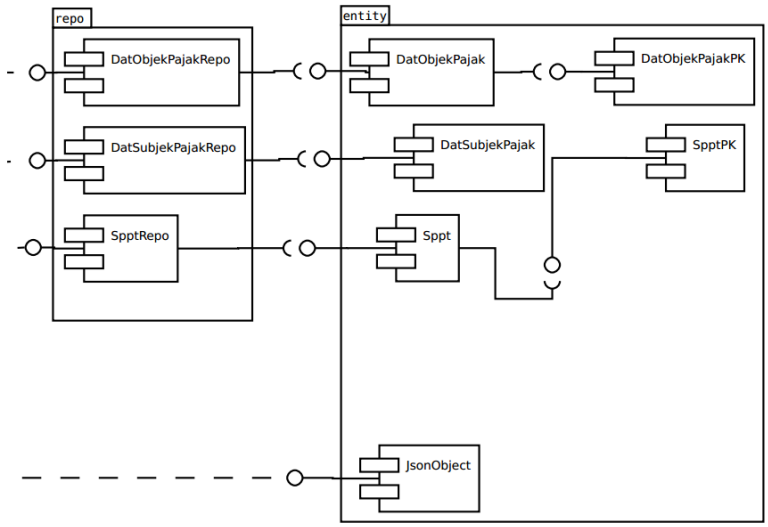
Gambar 2.5: Diagram *Component* Untuk Ujung-Depan (*frontend*)

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

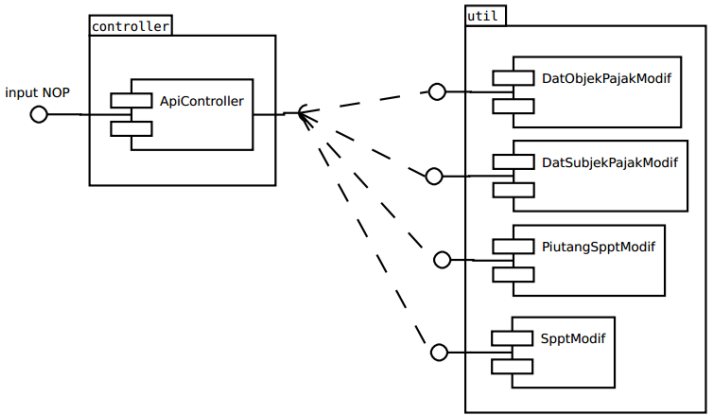
Diagram *component* yang membentuk bagian ujung belakang (*backend*) adalah seperti pada gambar 2.6, 2.7, dan 2.8 berikut ini :



Gambar 2.6: Diagram *Component* Bagian 1



Gambar 2.7: Diagram *Component* Bagian 2



Gambar 2.8: 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 2.7.

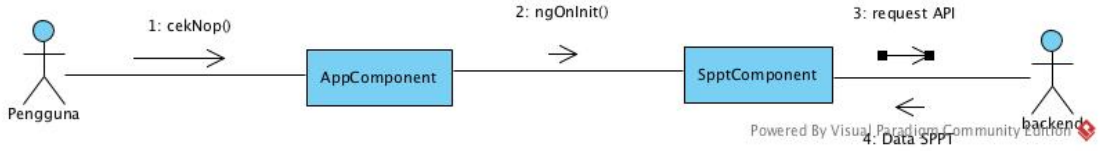
Kemudian tiap *service* akan melakukan pemanggilan data terhadap *repository*nya masing-masing yang kemudian akan dikembalikan dalam bentuk objek sesuai tabel yang diaksesnya seperti pada gambar 2.6 dan 2.7.

Hasil yang dikembalikan adalah beberapa objek dari kelas pada paket `util` seperti disebutkan pada gambar 2.8

2.4 Diagram *Communiation*

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 reffig:comm-dia-fe berikut ini :

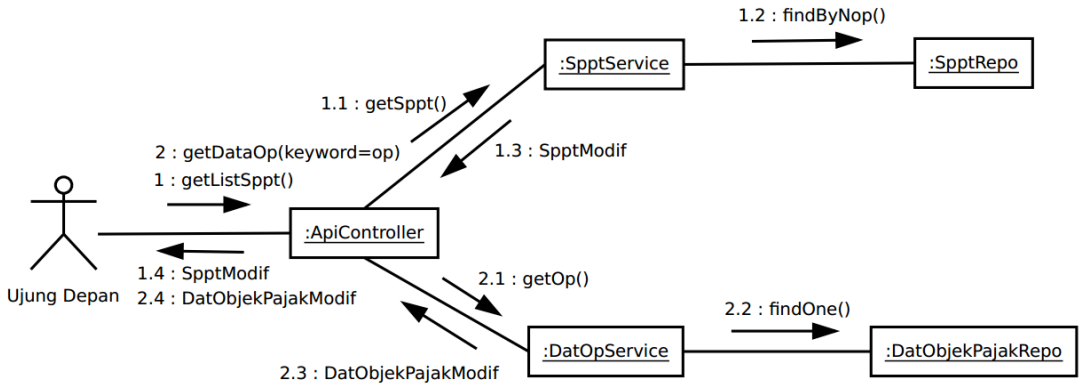


Gambar 2.9: Diagram *Communication* Untuk Bagian Ujung-Depan (*frontend*)

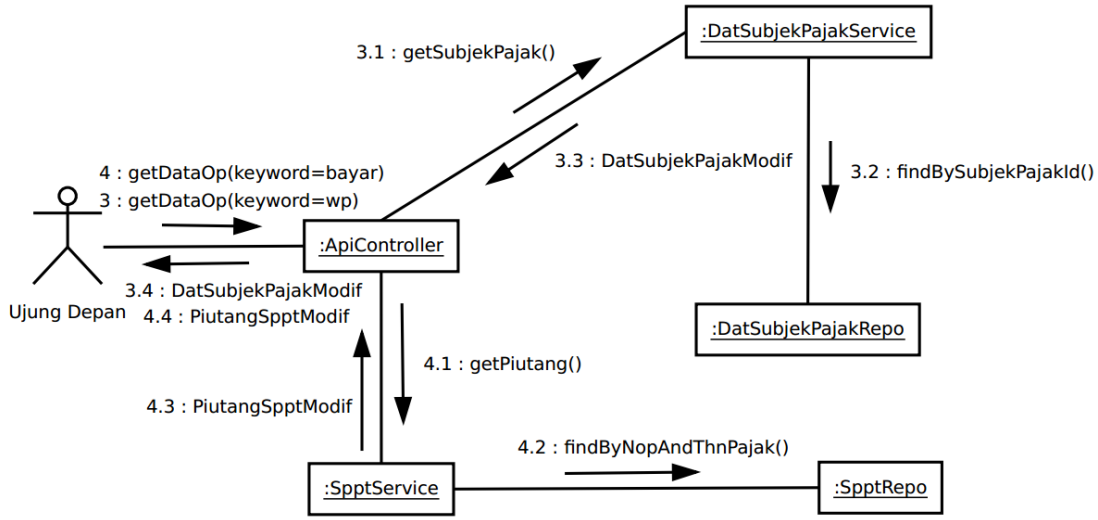
Alurnya adalah dari pengguna akan memasukkan Nomor Objek Pajak (NOP) pada kolom yang disediakan pada `AppComponent`, kemudian komponen ini akan memanggil fungsi `cekNop()` dan mengaktifkan `SpptComponent` yang secara otomatis akan memanggil fungsi `ngOnInit()`.

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 2.10 dan **??** berikut ini :



Gambar 2.10: Diagram *Communication* Bagian 1



Gambar 2.11: 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 adalah 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 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`, proesnya 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*).

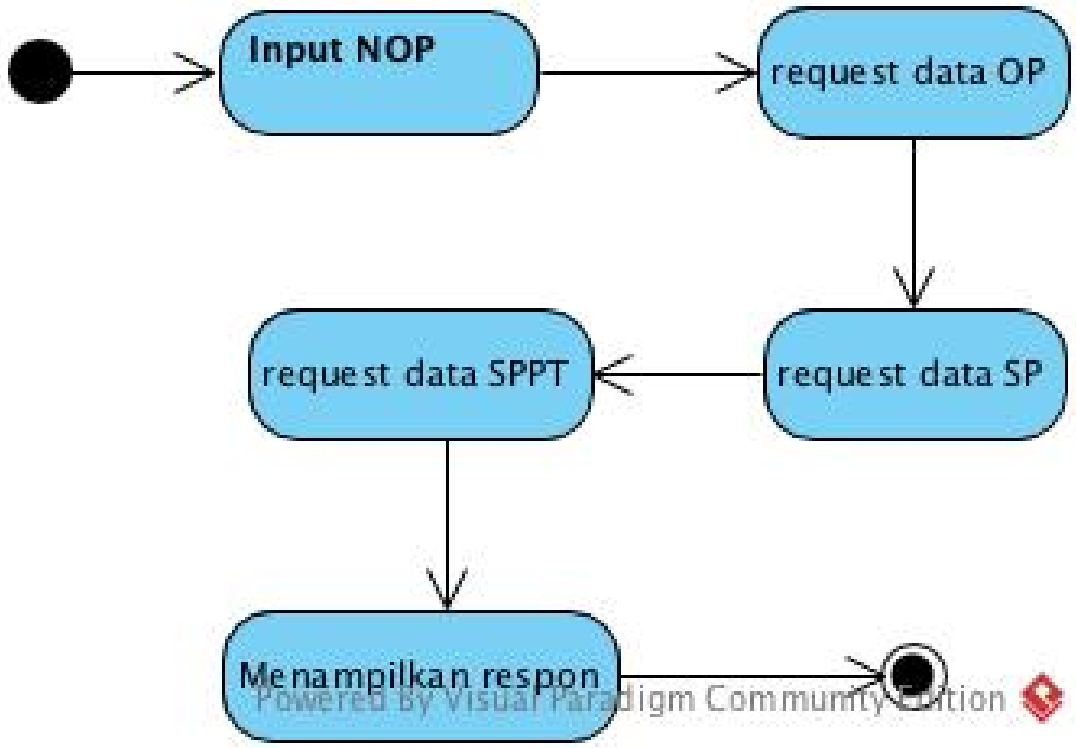
Skenario ke-4 sama saja seperti skenario ke-3 dan ke-2, akan memanggil *method* `getDataOp()` dengan parameter *keyword* berisi teks `bayar`. Langkah selan-

jutnya 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 (*frontend*) berupa objek dari kelas `PiutangSpptModif` dalam bentuk JSON.

2.5 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 2.12 berikut ini :

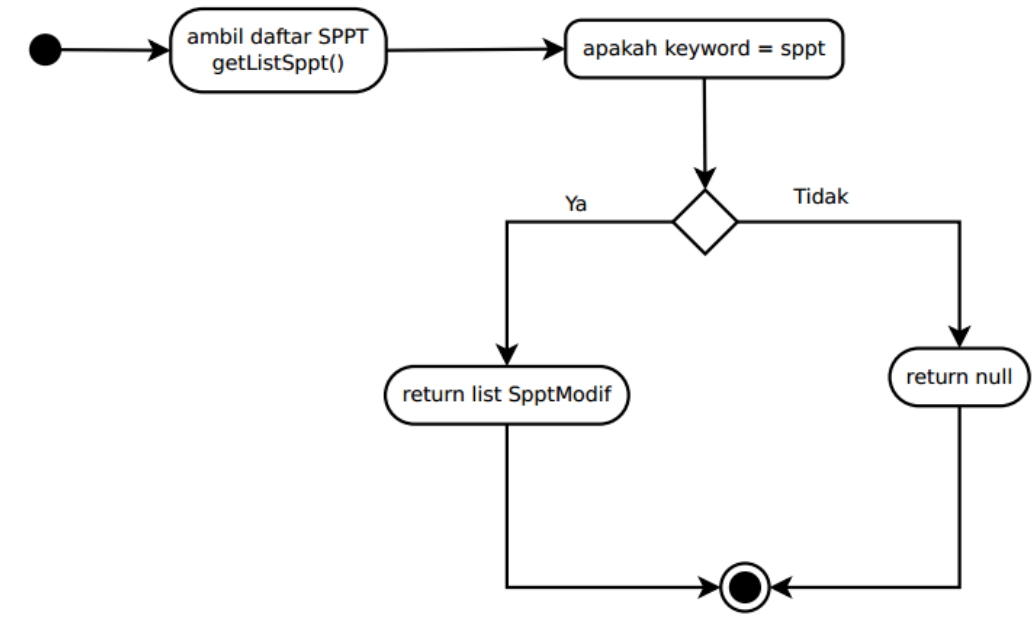


Gambar 2.12: Diagram *Activity* Untuk Bagian Ujung-Depan (*frontend*)

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

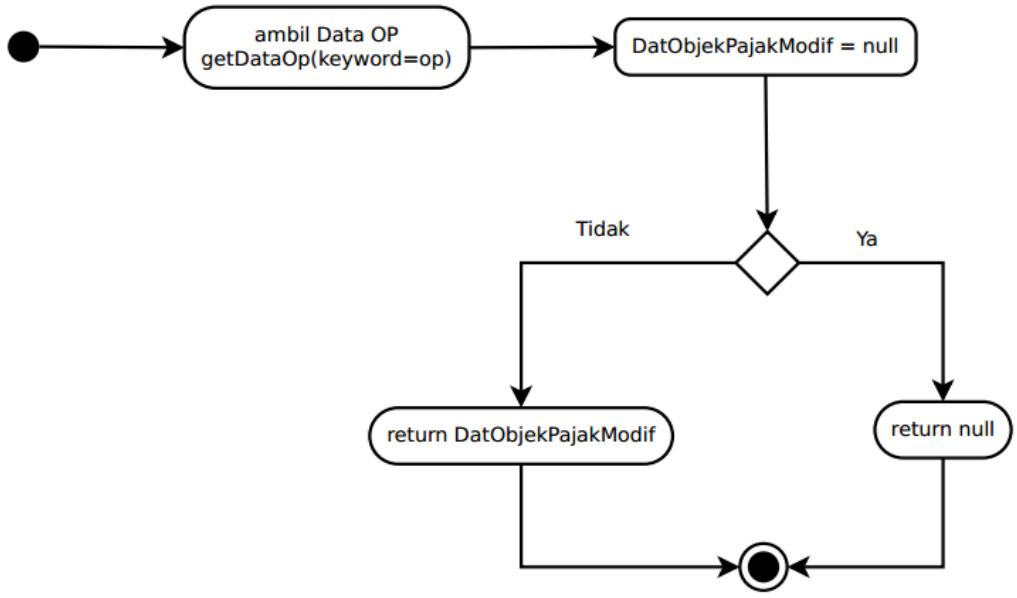
Pada skenario pertama, adalah aktivitas yang terjadi ketika ada *request* daftar tagihan pajak bumi dan bangunan sektor perdesaan dan perkotaan dari klien, diagram *activity* untuk skenario ini adalah seperti pada gambar 2.13 berikut ini :



Gambar 2.13: Diagram *Activity* Untuk Ambil Daftar Tagihan

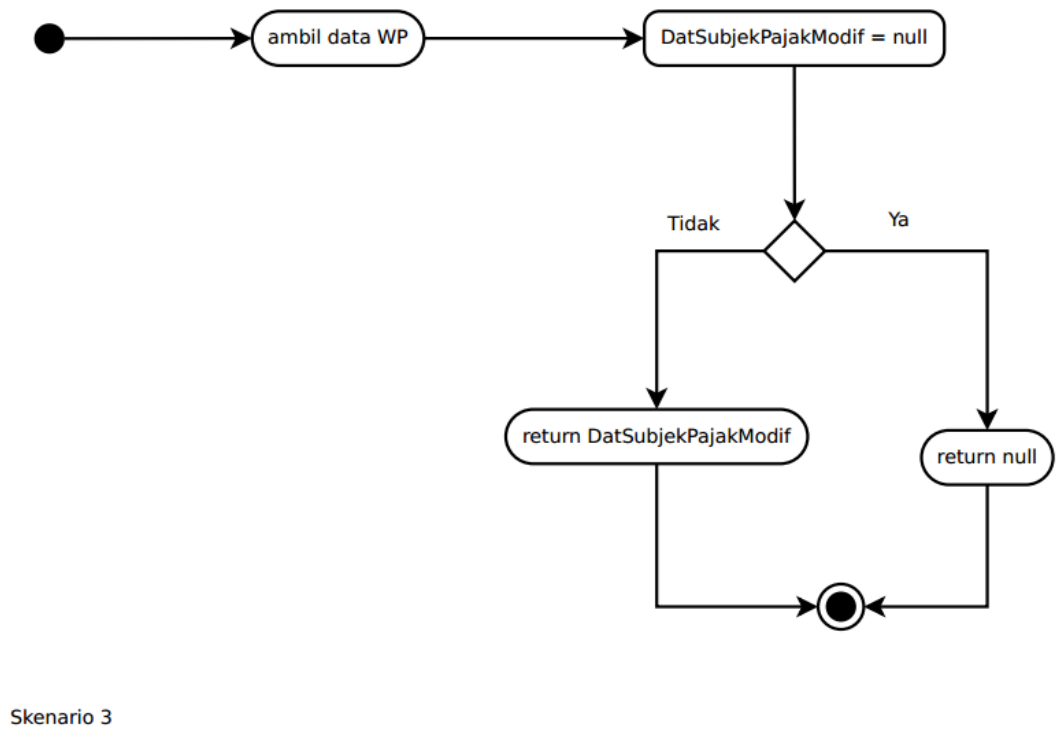
Diagram tersebut menunjukkan apabila data yang diminta oleh klien tidak ada pada sistem basis data, maka sistem akan mengembalikan nilai `null` atau kosong.

Skenario yang kedua terjadi ketika ada *request* data objek pajak dari klien. Diagram *activity* untuk skenario kedua ini seperti terlihat pada gambar 2.14 berikut ini :



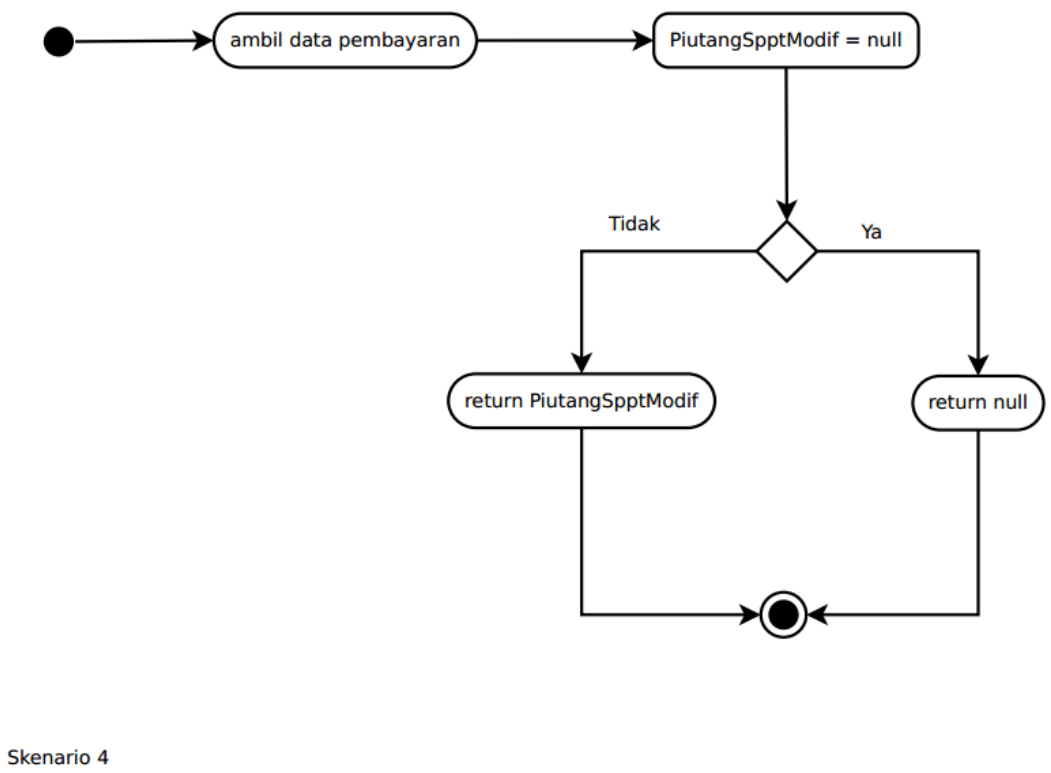
Gambar 2.14: Diagram *Activity* Skenario Kedua

Skenario berikutnya adalah skenario ketiga, yang terjadi ketika ada *request* data wajib pajak dari klien. Diagram *activity* untuk skenario ini adalah seperti pada gambar 2.15 berikut ini :



Gambar 2.15: Diagram *Activity* Skenario Ketiga

Skenario keempat terjadi ketika ada *request* data pembayaran dari klien. Diagram *activity* untuk skenario ini adalah seperti pada gambar 2.16 berikut ini :

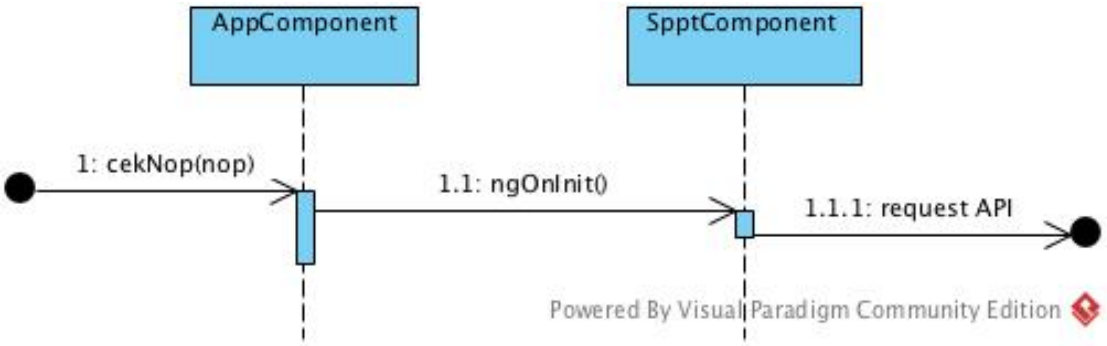


Gambar 2.16: Diagram *Activity* Skenario Keempat

2.6 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 2.17 berikut ini :



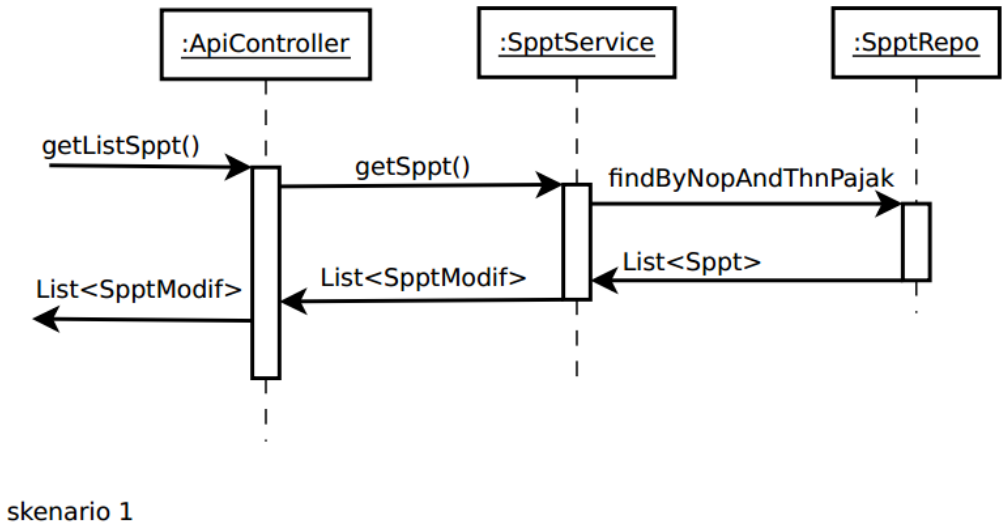
Gambar 2.17: Diagram *Sequence* Untuk Bagian Ujung-Depan (*frontend*)

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 (*Application Programmable Interface*) pada peladen bagian ujung belakang (*backend*).

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

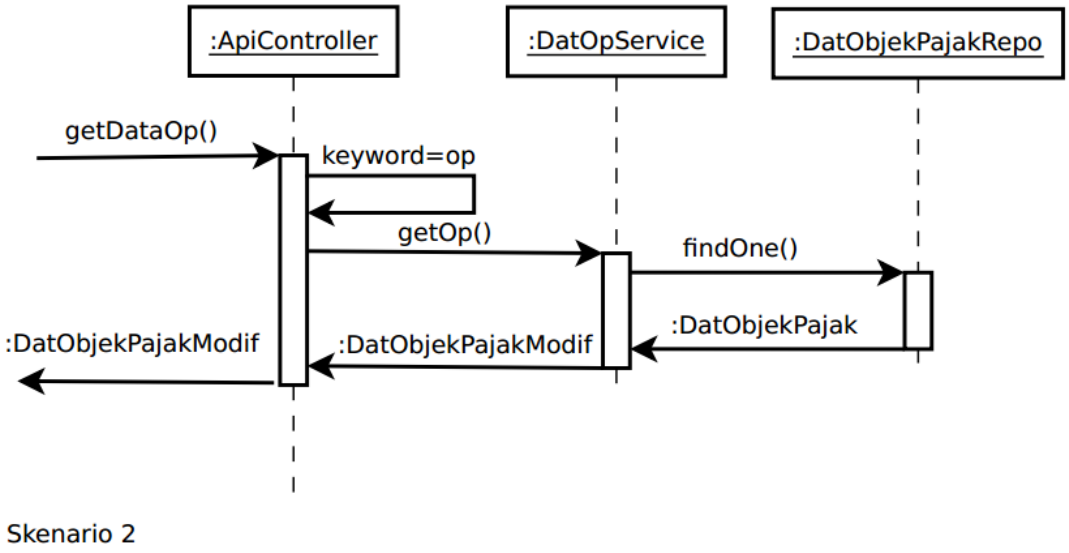
Pada skenario pertama, sistem akan memberikan daftar tagihan pajak bumi dan bangunan sektor perdesaan dan perkotaan untuk seluruh wajib pajak berdasarkan Nomor Objek Pajak (NOP) yang diminta.

Diagram *sequence* untuk skenario pertama ini seperti pada gambar 2.18 berikut ini :



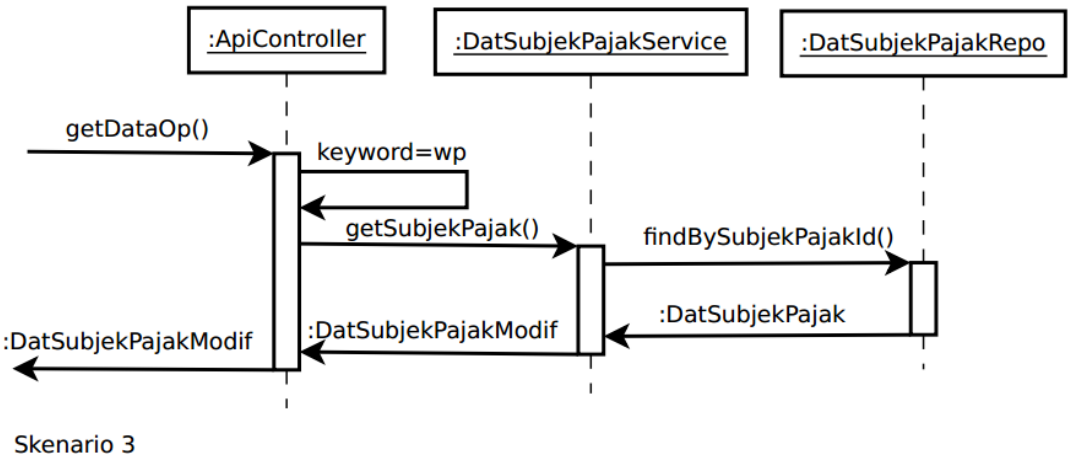
Gambar 2.18: Diagram *Sequence* Untuk Skenario Pertama

Untuk skenario kedua, sistem akan memberikan informasi mengenai objek pajak berdasarkan Nomor Objek Pajak (NOP) yang diminta. Diagram *sequence* untuk skenario ini adalah seperti pada gambar 2.19 berikut ini :



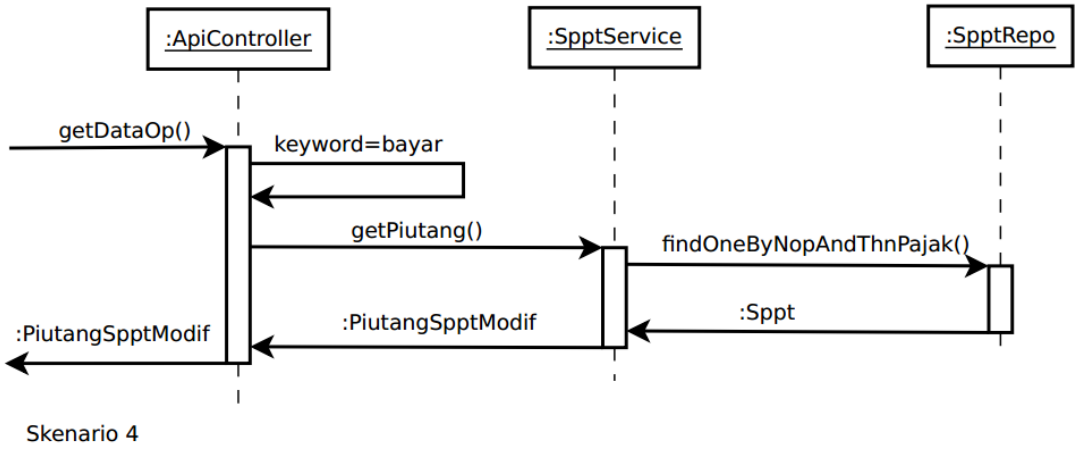
Gambar 2.19: Diagram *Sequence* Untuk Skenario Kedua

Untuk skenario ketiga, sistem akan memberikan informasi mengenai wajib pajak berdasarkan Nomor Objek Pajak (NOP) yang diminta. Diagram *sequence* untuk skenario ini adalah seperti pada gambar 2.20 berikut ini :



Gambar 2.20: Diagram *Sequence* Untuk Skenario Ketiga

Untuk skenario keempat, sistem akan memberikan tanggal jatuh tempo dan denda (apabila ada karena keterlambatan pembayaran) untuk objek pajak berdasarkan Nomor Objek Pajak (NOP) dan tahun pajak tertentu. Diagram *sequence* untuk skenario ini adalah seperti pada gambar 2.21 berikut ini :

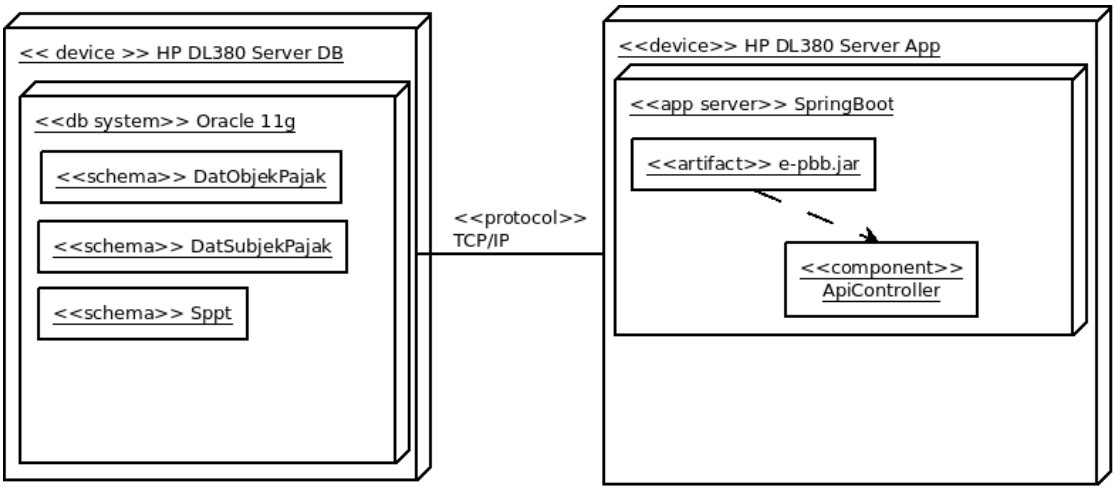


Gambar 2.21: Diagram *Sequence* Untuk Skenario Keempat

2.7 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 melayani pengguna aslinya.

Diagram ini digambarkan seperti pada gambar 2.22 berikut ini :



Gambar 2.22: 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.