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

5 Maret 2018

Priyanto Tamami, S.Kom.

1 TUJUAN SISTEM

Tujuan dari dibangunnya sistem informasi pembayaran Pajak Bumi dan Bangunan Sektor Perdesaan dan Perkotaan ini adalah untuk menjadi sarana komunikasi kepada wajib pajak serta melakukan kontrol bersama terhadap setoran pajak yang dibayarkan melalui petugas Desa/Kelurahan.

2 PEMODELAN SISTEM

Sistem akan terbagi menjadi 2 (dua) bagian, yaitu bagian ujung belakang (*backend*) dan bagian ujung depan (*frontend*), keduanya akan berkomunikasi dengan JSON.

Sistem informasi yang akan dibangun sangat sederhana, yaitu hanya menerima sebuah masukkan berupa Nomor Objek Pajak dan menampilkan informasi objek pajak, subjek pajak, dan besaran pajak terhutang, prosesnya yaitu bagian ujung depan (*frontend*) akan melakukan *request* ke peladen *REST* pada ujung belakang (*backend*), yang kemudian peladen *REST* akan melakukan respon terhadap data yang diminta.

Keseluruhan prosesnya akan dimodelkan dengan *Unified Modeling Language* (UML) seperti 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.

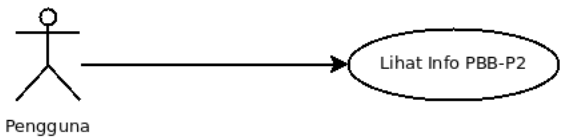
4. Diagram *Sequence*

   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 :



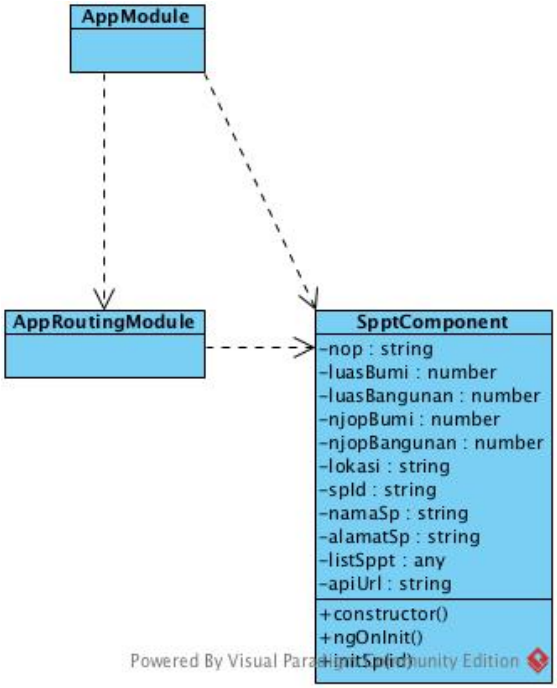
Gambar 1: Diagram *Use-Case* Sistem Pembayaran PBB-P2

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 2.2 berikut ini :



Gambar 2: Diagram *Class* Untuk Bagian Ujung Depan (*Frontend*)

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

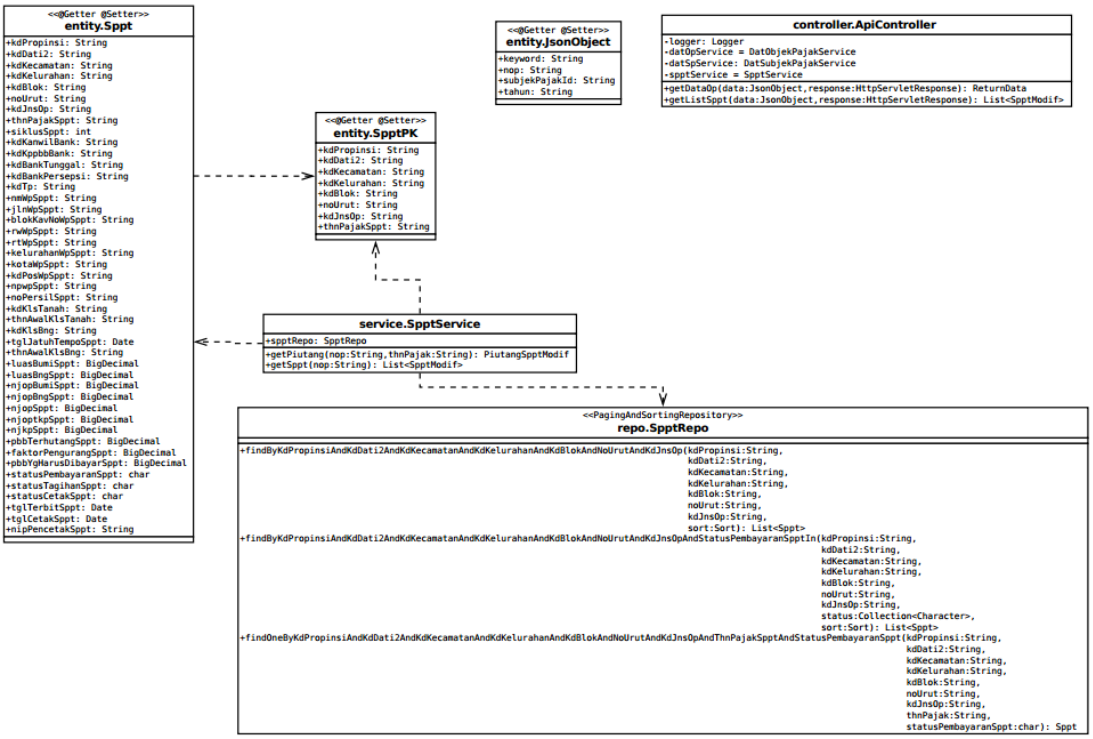
`AppModule`, kelas `AppRoutingModule` yang akan mengatur bagaimana tiap komponen kelas (dalam hal ini halaman) akan berganti-ganti sesuai dengan kejadian yang diinginkan oleh pengguna.

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`.

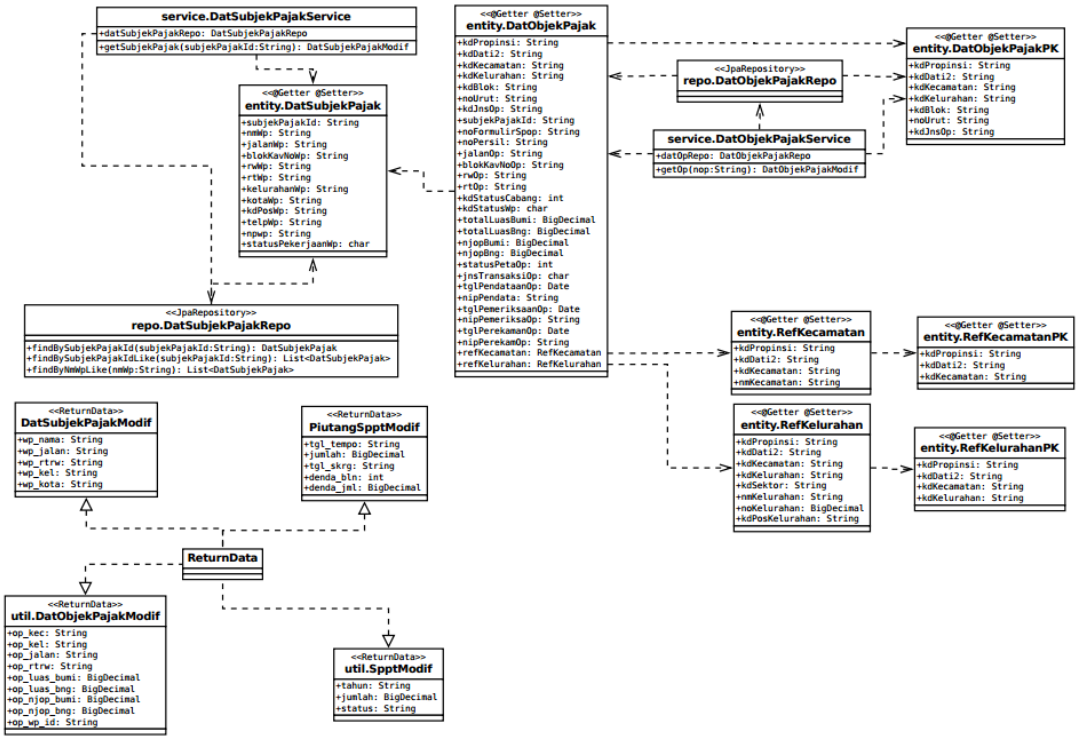
Kelas `AppComponent` sebetulnya kelas yang bertugas menjadi tampilan utama aplikasi, artinya begitu ada *browser* / peramban yang melakukan akses, halaman dari kelas `AppComponent` ini akan tampil terlebih dahulu menyambut pengguna.

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 3 dan 4 berikut ini :



Gambar 3: Diagram *Class* Untuk Ujung Belakang (*backend*) Bagian 1



Gambar 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_OBJECT_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 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 sebagaimaan mestinya.

2.3 Diagram *Component*

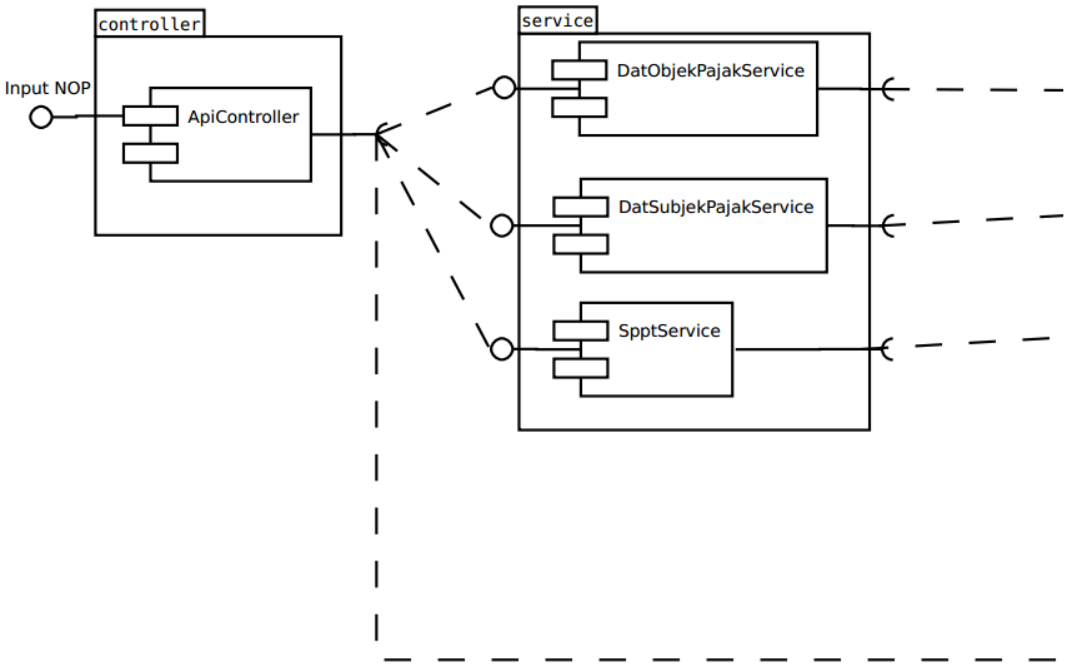
Diagram ini memberikan gambaran hubungan antar komponen, komponen mana yang membutuhkan data dan komponen mana yang memberikan data akan terlihat jelas pada diagram komponen ini.

Diagram *component* untuk bagian ujung-depan (*frontend*) ditunjukkan seperti pada gambar 5 berikut ini :

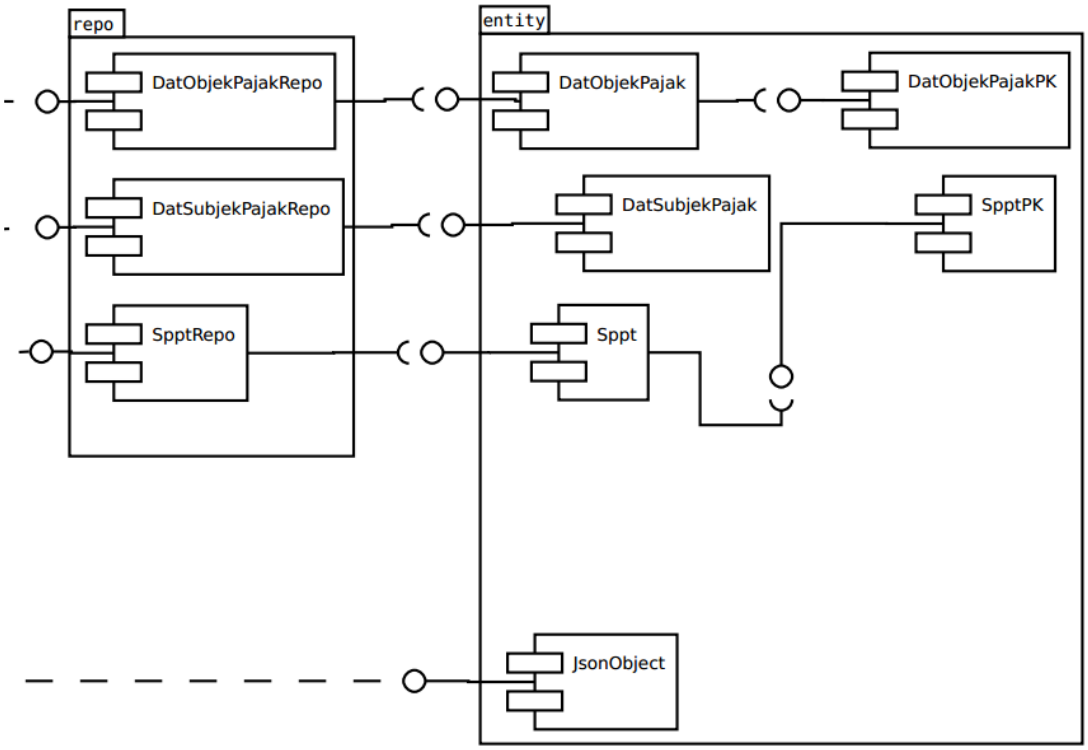
Gambar 5: Diagram *Component* Untuk Ujung-Depan (*frontend*)

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.

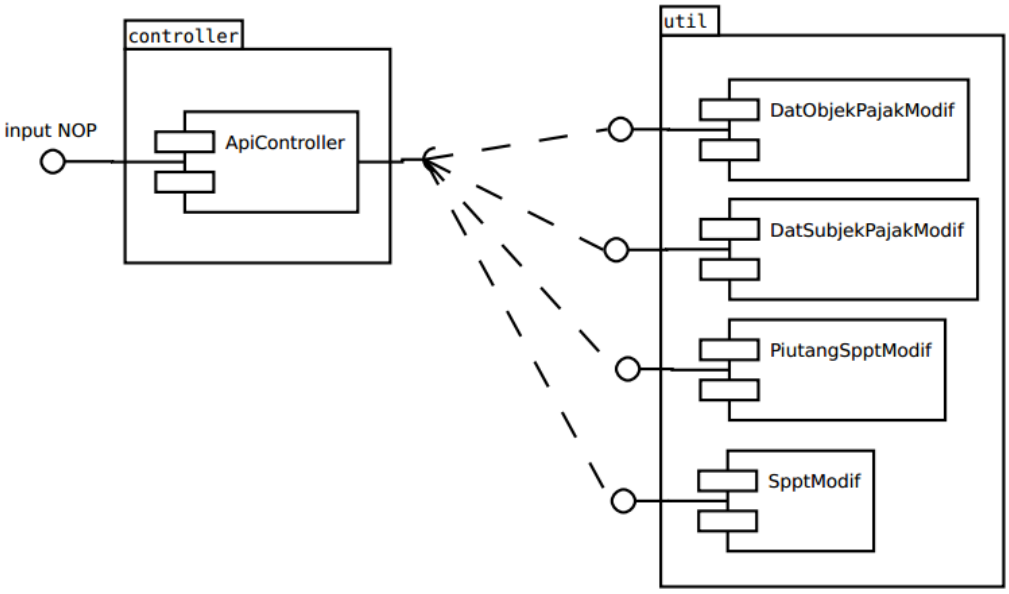
Diagram *component* yang membentuk bagian ujung belakang (*backend*) adalah seperti pada gambar 6, 7, dan 8 berikut ini :



Gambar 6: Diagram *Component* Bagian 1



Gambar 7: Diagram *Component* Bagian 2



Gambar 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.

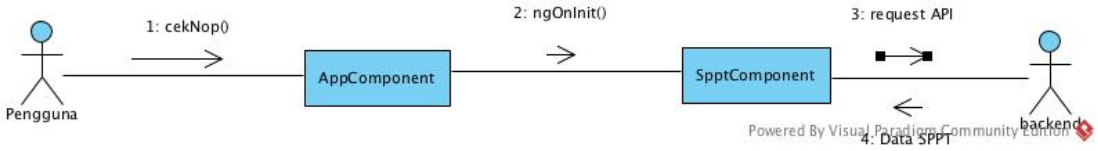
Kemudian tiap *service* akan melakukan pemanggilan data terhadap *repository*-nya masing-masing yang kemudian akan dikembalikan dalam bentuk objek sesuai tabel yang diaksesnya seperti pada gambar 6 dan 7.

Hasil yang dikembalikan adalah beberapa objek dari kelas pada paket `util` seperti disebutkan pada gambar 8.

2.4 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 9 berikut ini :

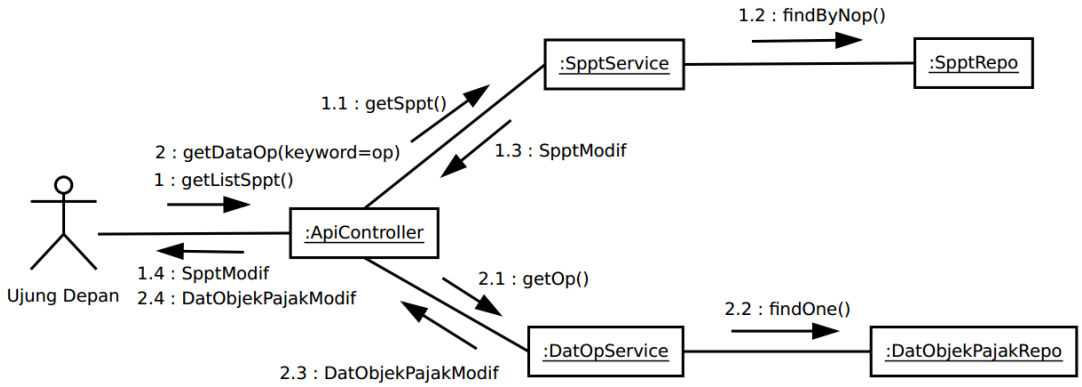


Gambar 9: Diagram *Communication* Untuk Bagian Ujung-Depan (*frontend*)

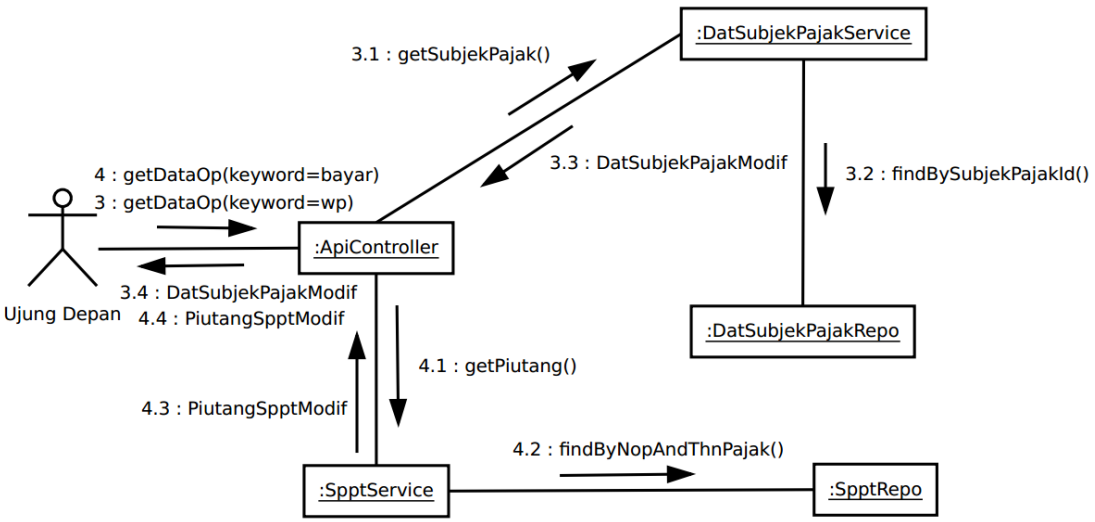
Alurnya adalah dari pengguna akan memasukkan Nomor Objek Pajak (NOP) pada kolom yang disediakan pada `AppComponent`, kemudian komponen ini akan memanggil fungsi `cekNop()` dan mengaktifkan `SpptComponent` yang secara otomatis akan memanggil fungsi `ngOnInit()`.

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 10 dan 11 berikut ini :



Gambar 10: Diagram *Communication* Bagian 1



Gambar 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 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 seknario ke-3 di gambar 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*).

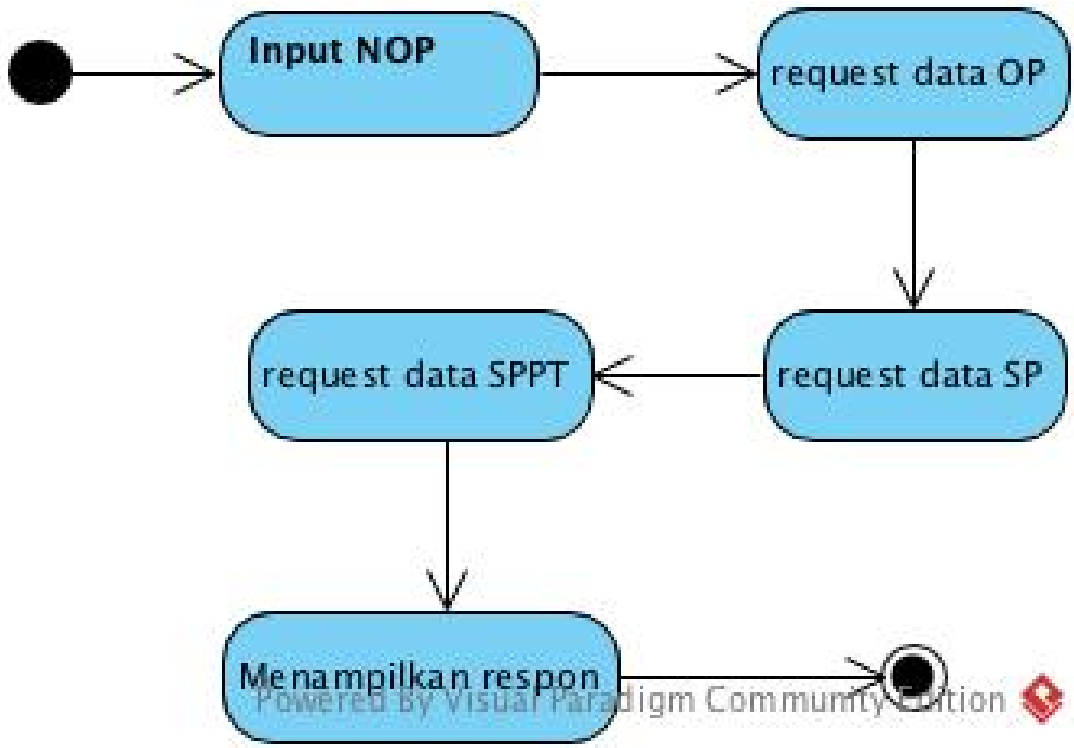
Skenario ke-4 sama saja seperti skenario ke-3 dan ke-2, akan memanggil *method* `getDataOp()` dengan parameter *keyword* berisi teks `bayar`. Langkah selanjutnya adalah memanggil *method* `getPiutang()` milik kelas `SpptService` yang kemudian akan mengambil data ke sistem basis data dengan memanggil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

an mengembalikan ke bagian ujung-depan (*front-end*) 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 12 berikut ini :



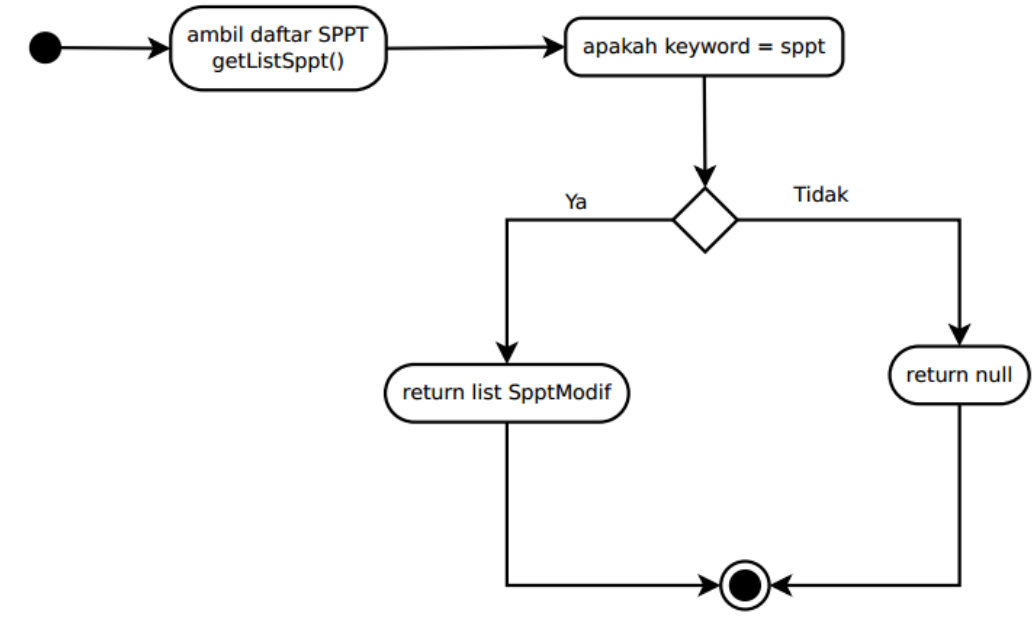
Gambar 12: Diagram *Activity* Untuk Bagian Ujung-Depan (*frontend*)

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

nampilkan 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 13 berikut ini :

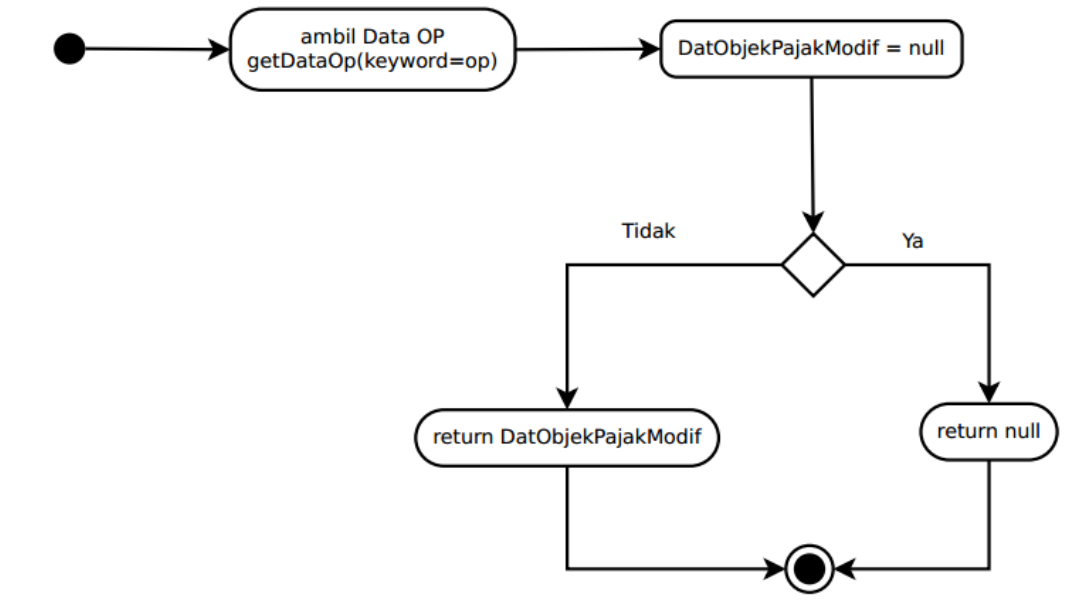


Gambar 13: Diagram *Activity* Untuk Ambil Daftar Tagihan

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

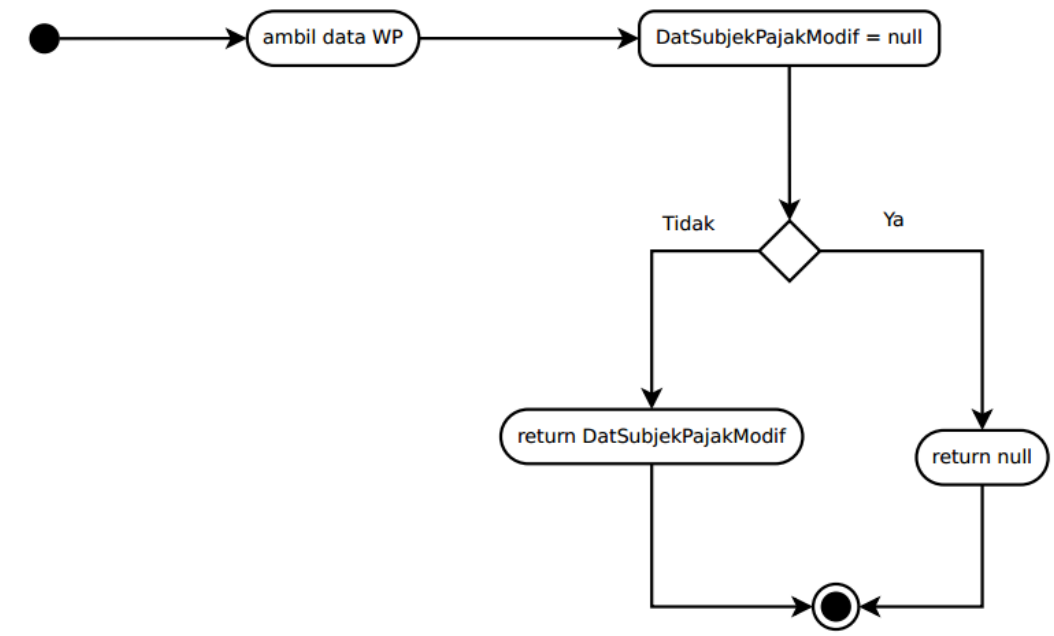
Skenario yang kedua terjadi ketika ada *request* data objek pajak dari klien. Diagram *activity* untuk skenario kedua ini seperti terlihat pada gambar 14 berikut

ini :



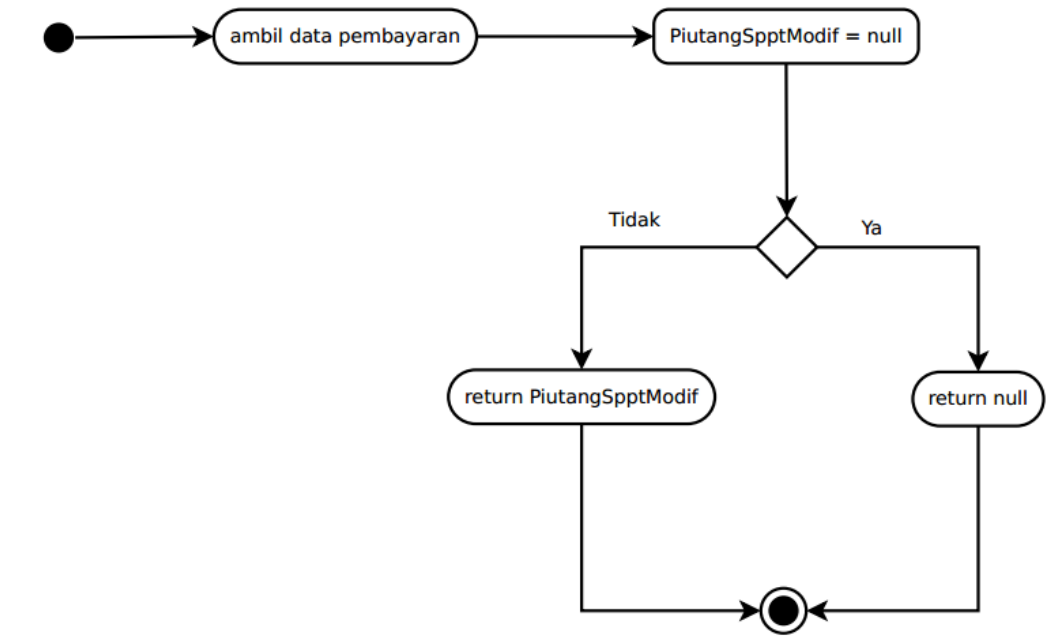
Gambar 14: Diagram *Activity* Skenario Kedua

Skenario berikutnya adalah skenario ketiga, yang terjadi ketika ada *request* data wajib pajak dari klien. Diagram *activity* untuk skenario ini adalah seperti pada gambar 15 berikut ini :



Gambar 15: Diagram *Activity* Untuk Skenario Ketiga

Skenario keempat terjadi ketika ada *request* data pembayaran dari klien. Diagram *activity* untuk skenario ini adalah seperti pada gambar 16 berikut ini :

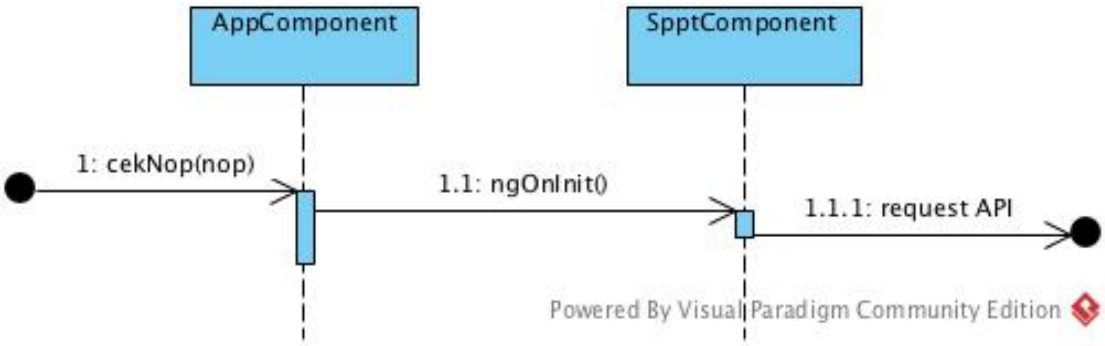


Gambar 16: Diagram *Activity* Untuk Skenario Keempat

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 17 berikut ini :



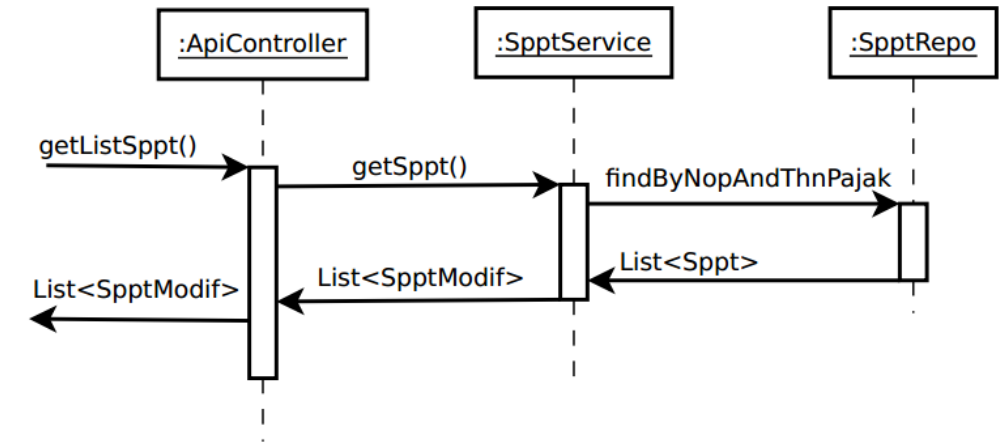
Gambar 17: Diagram *Sequence* Untuk Bagian Ujung-Depan (*frontend*)

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.

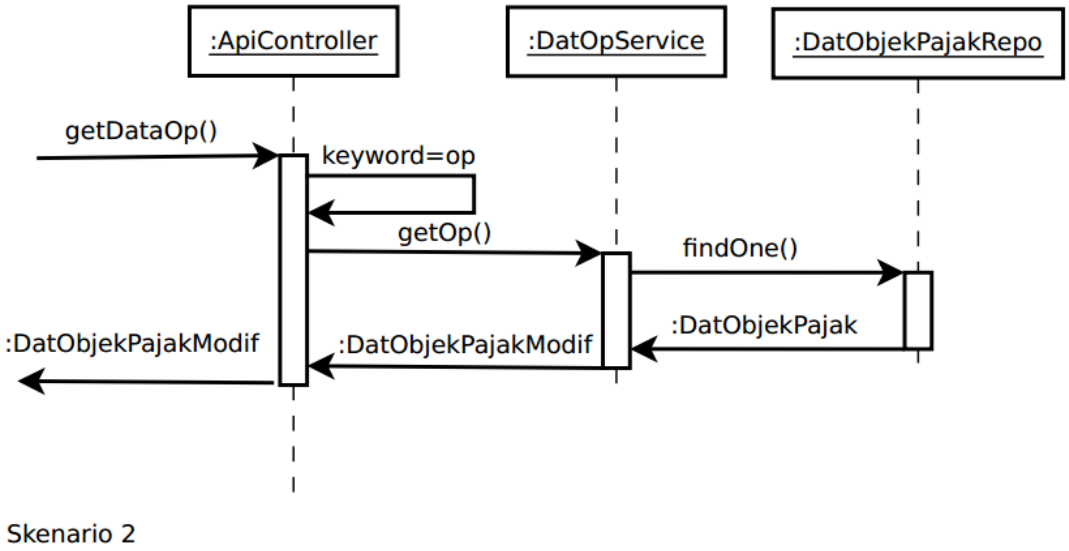
Pada skenario pertama, sistem akan memberikan daftar tagihan pajak bumi dan bangunan sektor perdesaan dan perkotaan untuk seluruh wajib pajak berdasarkan nomor objek pajak yang diminta.

Diagram *sequence* untuk skenario pertama ini seperti pada gambar 2.6 berikut ini :



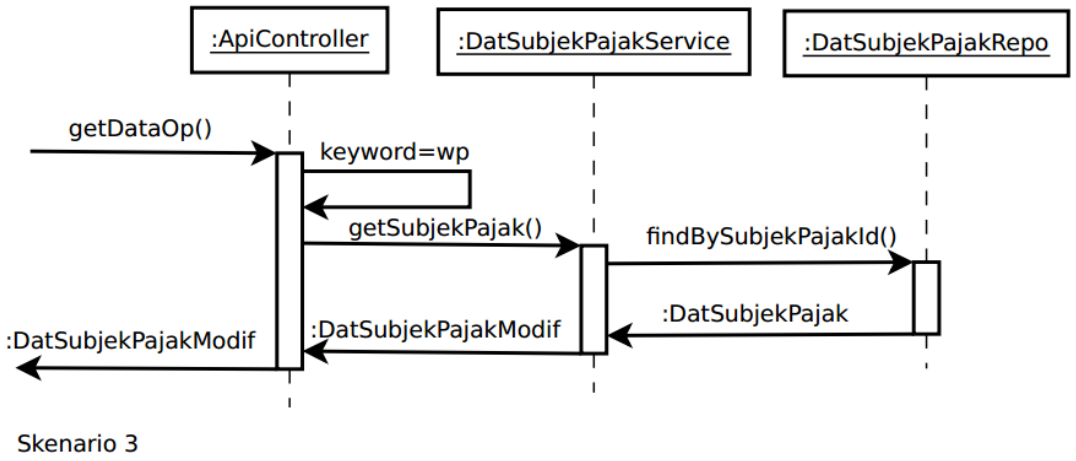
Gambar 18: Diagram *Sequence* Untuk Skenario Pertama

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 2.6 berikut ini :



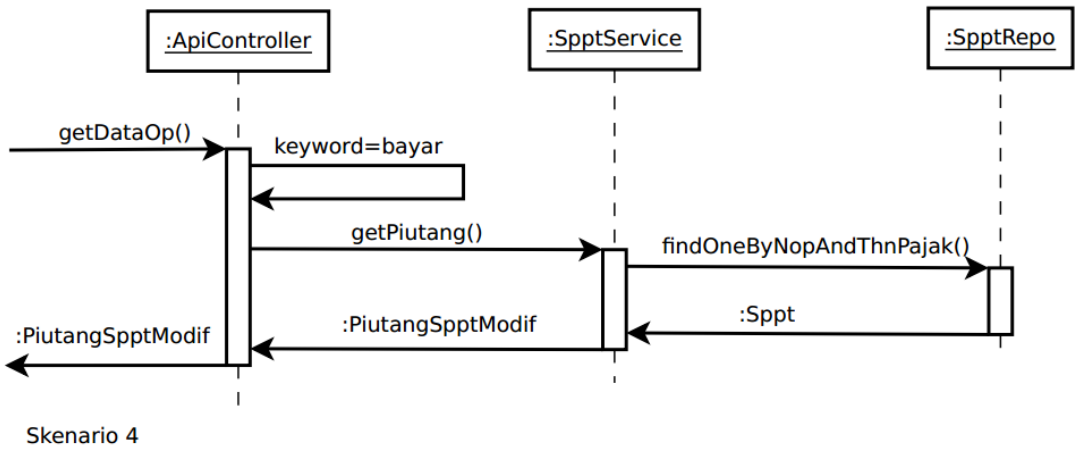
Gambar 19: Diagram *Sequence* Untuk Skenario Kedua

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 2.6 berikut ini :



Gambar 20: Diagram *Sequence* Untuk Skenario Ketiga

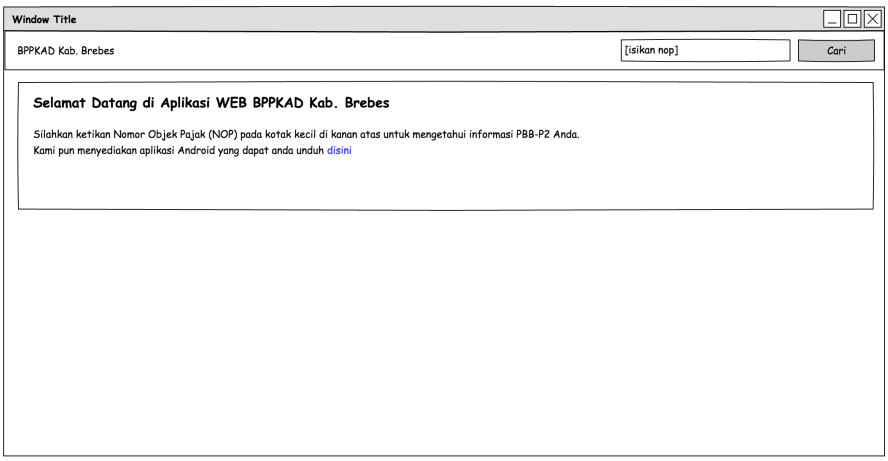
Untuk skenario keempat, sistem akan memberikan informasi tanggal jatuh tempo dan denda (apabila ada karena keterlambatan pembayaran) untuk objek pajak berdasarkan nomor objek pajak dan tahun pajak tertentu. Diagram *sequence* untuk skenario ini adalah seperti pada gambar 2.6 berikut ini :



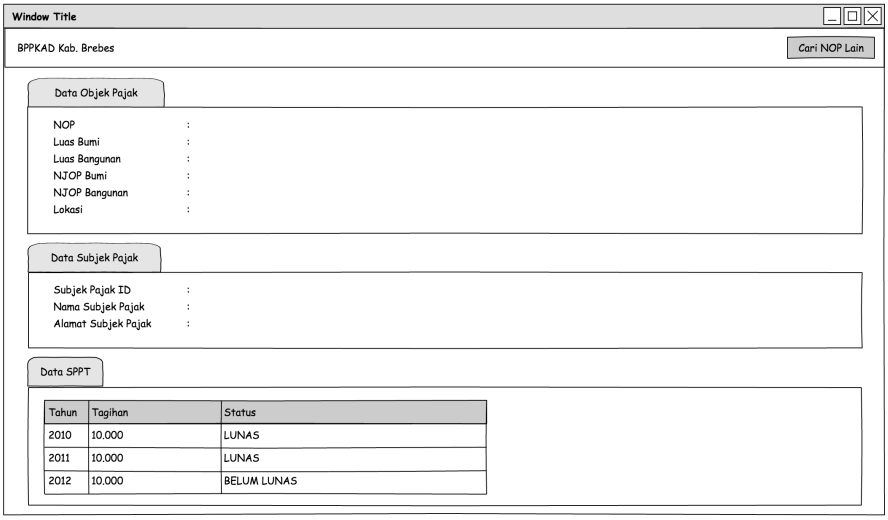
Gambar 21: Diagram *Sequence* Untuk Skenario Keempat

2.7 Desain Tampilan Aplikasi

Desain tampilan aplikasi yang diharapkan nantinya adalah seperti pada gambar 22 dan 23 berikut :



Gambar 22: Tampilan Awal Aplikasi



Gambar 23: Tampilan Aplikasi Setelah Memperoleh Hasil

3 AKTIVITAS PEMROSESAN DATA

3.1 Hasil Keluaran Yang Diharapkan

Keluaran yang diharapkan dari sistem informasi ini adalah tersedianya informasi mengenai data objek pajak seperti luas bumi, luas bangunan, nilai jual objek pajak untuk bumi, nilai jual objek pajak untuk bangunan, serta alamat dari objek pajak yang dicari.

Kemudian tersedianya informasi mengenai subjek pajak, seperti nomor identitas subjek pajak, nama subjek pajak, serta alamat tempat tinggal subjek pajak.

Informasi yang tidak kalah pentingnya adalah adanya informasi mengenai tahun pajak, besaran pajak yang terhutang, serta status pembayarannya, apakah sudah lunas atau belum.

3.2 Identifikasi Data Masukkan

Data atau informasi yang digunakan untuk melakukan proses pencarian data ini hanya berdasarkan Nomor Objek Pajak (NOP) yang tertera pada bagian kiri atas lembar Surat Pemberitahuan Pajak Terhutang (SPPT) yang diterbitkan setiap tahunnya.

3.3 Alur Proses Data

Alur proses datanya sebetulnya sudah dimodelkan dengan grafik UML pada bagian sebelumnya, yang secara garis besar bahwa alurnya adalah pertama pengguna sistem informasi / aplikasi akan memasukkan informasi berupa Nomor Objek Pajak (NOP) ke dalam kolom yang sudah disediakan.

Data NOP ini akan dikirimkan dari bagian ujung-depan (*frontend*) ke bagian ujung-belakang (*backend*) dengan mengirimkan objek JSON dengan bagian ujung-belakang (*backend*) sebagai peladen REST.

Kemudian peladen pada bagian ujung-belakang (*backend*) akan melakukan pencarian pada sistem basis data berdasarkan informasi NOP yang telah diterima, apabila data ditemukan, maka informasinya akan dikirimkan ke bagian ujung-depan (*frontend*) dan disusun sedemikian rupa sehingga pengguna dapat menangkap informasi yang disajikan. Namun apabila data yang dicari tidak ada di dalam sistem basis data, maka aplikasi akan menampilkan informasi kosong bahwa data yang dicari tidak dapat ditemukan.