

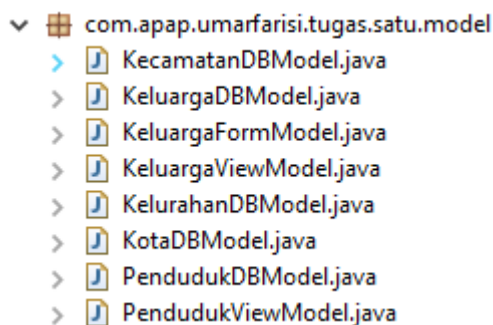
## Proses Pengembangan Tugas 1

Saya mengembangkan tugas ini dengan beberapa tahap. Berikut ini tahap serta subtahap yang dilakukan untuk mengembangkan tugas ini :

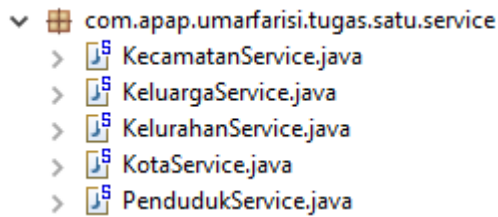
1. Membuat *back-end* pada setiap fitur. Saya membuat code untuk *back-end* pada setiap fitur yang ada. Pertama membuat kelas – kelas lapisan Service. Berbarengan dengan membuat lapisan Service, saya membuat kelas – kelas Mapper dan kelas – kelas pada lapisan Model.
2. Setelah itu saya membuat kelas – kelas pada lapisan Controller. Untuk memvalidasi data – data yang diberikan kelas Service kepada kelas Controller, saya membuat *log* untuk melihat data – data tersebut sebelum di kelola di lapisan View.
3. Selanjutnya membuat *layout* – *layout* pada lapisan View. Data – data yang diberikan dari kelas Controller ke *layout* – *layout* tersebut dikelola untuk ditampilkan di lapisan View tersebut.
4. Selanjutnya saya memberikan optimasi *database*, dan melakukan *testing* pada *project*.
5. Selanjutnya saya memberikan *validation* untuk setiap Form Post.
6. Terakhir, saya *testing project*.

## Struktur Project, Package, dan Implementasi MVC

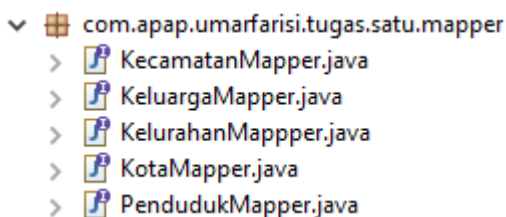
Struktur *project* pada tugas ini berdasarkan *template* dari Spring, dan struktur package yang saya lakukan pada tugas ini berdasarkan *pattern* Model View Controller (MVC). Pada tugas ini, saya mengimplementasi *pattern* MVC yaitu dengan membagi *project* menjadi beberapa lapisan (*layer*). Lapisan tersebut terdiri dari lapisan Model, lapisan View, dan lapisan Controller. Pada tugas ini, lapisan Model berisi kelas – kelas yang merepresentasikan data yang ingin digunakan pada *project* yaitu data untuk ditampilkan di *view*, data untuk menyimpan hasil *input* pengguna di sebuah *form*, dan data untuk merepresentasikan tabel di database. Kelas – kelas tersebut ditempatkan di satu *package* yang sama agar mudah diatur yaitu *package* `com.apap.umarfarisi.tugas.satu.model` dan berikut ini *screenshot*-nya :



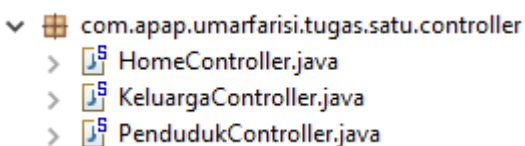
Lapisan Service berisi kelas – kelas yang merepresentasikan *business logic*. Kelas – kelas tersebut dibagi berdasarkan data – data utama yaitu Penduduk, Keluarga, Kelurahan, Kecamatan, dan Kota sehingga terdapat 5 kelas *service* yaitu *PendudukService*, *KeluargaService*, *KelurahanService*, *KecamatanService*, dan *KotaService*. Contohnya *business logic* untuk menampilkan daftar penduduk akan diletakan di kelas *PendudukService*. Kelas – kelas tersebut ditempatkan di satu *package* yang sama agar mudah diatur yaitu *package* `com.apap.umarfarisi.tugas.satu.service` dan berikut ini *screenshot*-nya :



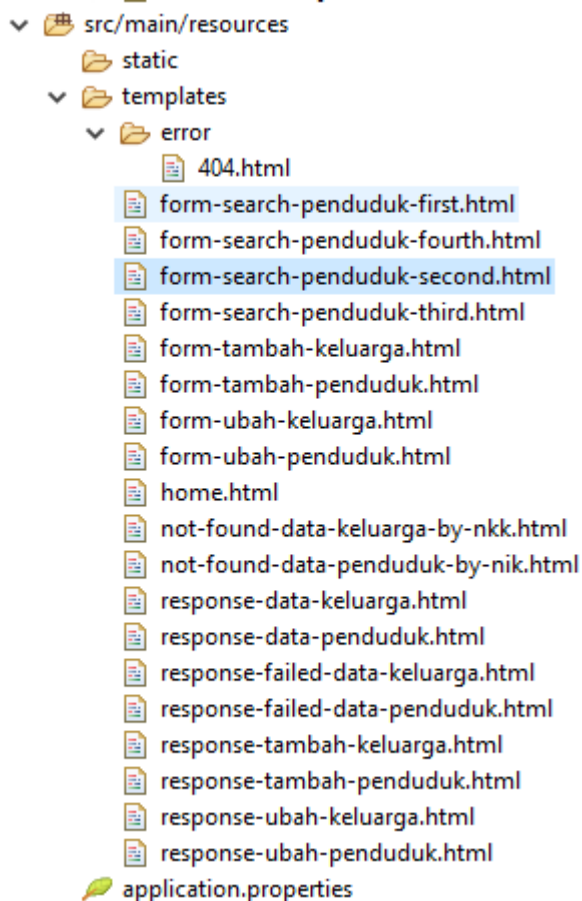
Selain kelas – kelas pada lapisan *service* tersebut, terdapat kelas – kelas yang memiliki fungsi yang menyerupai kelas *service* yaitu sebagai *business logic*. Kelas – kelas dipanggil kelas *mapper*. Perbedaan kelas – kelas *service* dan *mapper* adalah kelas – kelas *mapper* langsung berhubungan dengan *database*, kelas – kelas *mapper* berisi *query – query* pada *database*, sedangkan kelas – kelas *service* tidak langsung berhubungan dengan *database* namun menggunakan kelas – kelas *mapper* untuk melakukan *create read update delete* (CRUD) pada data di *database*. Kelas – kelas *mapper* dibagi berdasarkan data – data yang ada di *database* yaitu Penduduk, Keluarga, Kelurahan, Kecamatan, dan Kota sehingga terdapat 5 kelas *mapper* yaitu PendudukMapper, KeluargaMapper, KelurahanMapper, KecamatanMapper, dan KotaMapper. Pembagian tersebut agar mudah mengatur operasi CRUD berdasarkan data – data yang ada di *database*. Seperti kelas – kelas di lapisan *service*, kelas – kelas di *mapper* ditempatkan di satu *package* yang sama agar mudah diatur yaitu *package* `com.apap.umarfarisi.tugas.satu.mapper` dan berikut ini *screenshot*-nya :



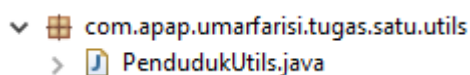
Lapisan Controller berisi kelas – kelas yang berfungsi untuk pengontrolan lapisan View. Kelas – kelas tersebut dibagi berdasarkan *endpoint* yang ada pada tugas Penduduk, Keluarga, dan Home sehingga terdapat 3 kelas *controller* yaitu PendudukController, KeluargaController, dan HomeController. Kelas – kelas tersebut ditempatkan di satu *package* yang sama agar mudah diatur yaitu *package* `com.apap.umarfarisi.tugas.satu.controller` dan berikut ini *screenshot*-nya :



Lapisan View berisi *layout – layout* bertipe Html yang berfungsi untuk berinteraksi dengan pengguna. *Layout – layout* tersebut terdapat pada `source/main/resource/templates`. Berikut ini *screenshot*-nya :



Selain kelas – kelas di setiap lapisan tersebut, terdapat kelas *utils* yang membuat *code* lebih *reuseable*, misalnya dapat digunakan di beberapa *service*. Pada tugas ini, saya membuat satu kelas *utils* yaitu PendudukUtils yang berisi *method* untuk *generate* nkk. *Method* tersebut digunakan di dua kelas *service* yaitu PendudukService dan KeluargaService. Untuk memudahkan pengaturan pada kelas *utils*, kelas *utils* ditempatkan pada satu *package* dengan prefix yang sama dengan kelas – kelas pada tiap lapisan, *package* tersebut adalah com.apap.umarfarisi.tugas.satu.utils. Berikut ini *screenshot*-nya :



## Optimasi Terhadap Database

Saya melakukan optimasi terhadap database dengan cara memberikan *primary key* dan *index* pada beberapa kolom pada tabel. Pada tabel Penduduk, saya menambahkan *primary key* pada kolom id dan menambahkan *index* pada kolom nik. Berikut ini *query* pada dbms yang saya lakukan untuk menambahkan hal tersebut di tabel Penduduk :

```
ALTER TABLE Penduduk ADD PRIMARY KEY (id);  
CREATE INDEX idx_penduduk_nik ON Penduduk (nik);
```

Pada tabel Keluarga, saya menambahkan *primary key* pada kolom id dan menambahkan *index* pada kolom nomor\_kk. Berikut ini *query* dbms yang saya lakukan untuk menambahkan hal tersebut di tabel Keluarga :

```
ALTER TABLE Keluarga ADD PRIMARY KEY (id);
CREATE INDEX idx_keluarga_nomor_kk ON Keluarga (nomor_kk);
```

Pada tabel Kelurahan, saya menambahkan *primary key* pada kolom id dan menambahkan dua *index* yaitu satu *index* pada kolom kode\_kelurahan dan yang satu *index* pada kolom id\_kecamatan . Berikut ini *query* dbms yang saya lakukan untuk menambahkan hal tersebut di tabel Kelurahan:

```
ALTER TABLE Kelurahan ADD PRIMARY KEY (id);
CREATE INDEX idx_kelurahan_kode_kelurahan ON Kelurahan (kode_kelurahan);
CREATE INDEX idx_kelurahan_id_kecamatan ON Kelurahan (id_kecamatan);
```

Pada tabel Kecamatan, saya menambahkan *primary key* pada kolom id dan menambahkan dua *index* yaitu satu *index* pada kolom kode\_kecamatan dan yang satu *index* pada kolom id\_kota . Berikut ini *query* dbms yang saya lakukan untuk menambahkan hal tersebut di tabel Kecamatan:

```
ALTER TABLE Kecamatan ADD PRIMARY KEY (id);
CREATE INDEX idx_kecamatan_kode_kecamatan ON Kecamatan (kode_kecamatan);
CREATE INDEX idx_kecamatan_id_kota ON Kecamatan (id_kota);
```

Pada tabel Kota, saya menambahkan *primary key* pada kolom id dan menambahkan *index* pada kolom kode\_kota. Berikut ini *query* dbms yang saya lakukan untuk menambahkan hal tersebut di tabel Kota :

```
ALTER TABLE Kota ADD PRIMARY KEY (id);
CREATE INDEX idx_kota_kode_kota ON Kota (kode_kota);
```

Kolom – kolom yang saya jadikan *primary key* atau *index* tersebut adalah kolom – kolom yang sering ada di *where clause* pada *query* saya di aplikasi. Kolom – kolom tersebut berada di *where clause* untuk perbandingan proses *join* antar tabel. Dengan menambahkan *primary key* atau *index* pada kolom – kolom tersebut, proses *join* antar tabel akan lebih cepat dilakukan.

Selain itu, saya menambahkan **AUTO INCREMENT** pada Penduduk(id), Keluarga(id), Kelurahan(id), Kecamatan(id), dan Kota(id)

## Eksperimen Stress Testing

Saya melakukan *stress testing* dengan menggunakan Jmeter. Berikut ini properti yang saya isi pada penggunaan Jmeter untuk *stress testing* :

- Number of Threads : 100
- Ramp – up Period : 10
- Loop Count : 1

### Fitur 1 Tampilkan Data Penduduk Berdasarkan NIK

GET : <http://localhost:8080/penduduk?nik=3101010102640001>

IP Request	1841	✓	147/1	147	1841
No of Samples	100	Latest Sample	1841	Average	2918
				Deviation	861

## Fitur 2 Tampilkan Data Keluarga Beserta Daftar Anggota Berdasarkan Nomor KK

GET : <http://localhost:8080/keluarga?nkk=3101010101050001>

Request	3510		2160	147	3510	
No of Samples	100		Latest Sample	3005	Average 3118	Deviation 1115

## Fitur 3 Menambahkan Penduduk Baru Sebagai Keluarga

GET : <http://localhost:8080/penduduk/tambah>

Request	21		2111	144	21	
No of Samples	100		Latest Sample	3	Average 3	Deviation 5

POST :

<http://localhost:8080/penduduk/tambah?nama=Badu+BuDiman&tempatLahir=Jombang+Jawa&tanggalLahir=2017-03-02&jenisKelamin=0&golonganDarah=AB+&agama=Islam&statusPerkawinan=belum+kawin+tapi+udah+ada+jodoh+asek&statusDalamKeluarga=anak&pekerjaan=bekerja+keras&wafat=1&idKeluarga=3>

IP Request	25		343	438	25
IP Request	27		343	438	27
No of Samples 100      Latest Sample 30      Average 32      Deviation 16					

## Fitur 4 Menambahkan Keluarga Baru

GET : <http://localhost:8080/keluarga/tambah/>

Request	2		872	134	2		
No of Samples	100	Latest Sample	2	Average	2	Deviation	4

POST :

<http://localhost:8080/keluarga/tambah?alamat=KOKOO+KIIKIKI&rt=123&rw=321&kelurahan=PULAU+TIDUNG&kecamatan=KEPULAUAN+SERIBU+SELATAN&kota=KABUPATEN+KEPULAUAN+SERIBU>

## Fitur 5 Mengubah Data Penduduk


GET : <http://localhost:8080/penduduk/ubah/3101010101690001>

TP Request	3		2044	148	3		
No of Samples	100	Latest Sample	4	Average	3	Deviation	0

POST :


<http://localhost:8080/penduduk/ubah/3101010101690001?nama=Baduuuuuu+BuDiman&tempatLahir=Jombang+Jawa&tanggalLahir=1969-01-01&jenisKelamin=0&golonganDarah=AB+&agama=Islam&statusPerkawinan=belum+kawin+tapi+ud>

[ah+ada+jodoh+asek&statusDalamKeluarga=anak&pekerjaan=bekerja+keras&wafat=1&idKeluarga=261](#)

Request	17		338	160	17		
No of Samples	100	Latest Sample	26	Average	38	Deviation	25

#### Fitur 6 Mengubah Data Keluarga

GET : <http://localhost:8080/keluarga/ubah/3101011607090001>

Request	URI		URI	URI	URI		
No of Samples	100	Latest Sample	82	Average	100	Deviation	76

POST :

<http://localhost:8080/keluarga/ubah/3101011607090001?alamat=HAHAHA+Dk.+Bara+No.+766&rt=057&rw=052&kelurahan=PULAU+TIDUNG&kecamatan=KEPULAUAN+SERIBU+SELATAN&kota=KABUPATEN+KEPULAUAN+SERIBU>

Request	7/10		338	352	7/10
No of Samples	100	Latest Sample	15253	Average 8516	Deviation 4321

#### Fitur 7 Mengubah Status Kematian Penduduk

POST : <http://localhost:8080/penduduk?nik=3101010102640001>

Request	6/15	338	454	7/10	
No of Samples	100	Latest Sample	6316	Average 6456	Deviation 1356

#### Fitur 8 Tampilkan Data Penduduk Berdasarkan Kota/Kabupaten, Kecamatan, dan Kelurahan Tertentu

GET : [http://localhost:8080/penduduk/cari?id\\_kota=1&id\\_kecamatan=1&id\\_kelurahan=1](http://localhost:8080/penduduk/cari?id_kota=1&id_kecamatan=1&id_kelurahan=1)

No of Samples	100	Latest Sample	5348	Average 5843	Deviation 1523
---------------	-----	---------------	------	--------------	----------------

#### Kesimpulan

Berdasarkan hasil yang *stress testing* yang didapatkan di atas, menurut saya aplikasi pada tugas 1 ini tidak lambat.