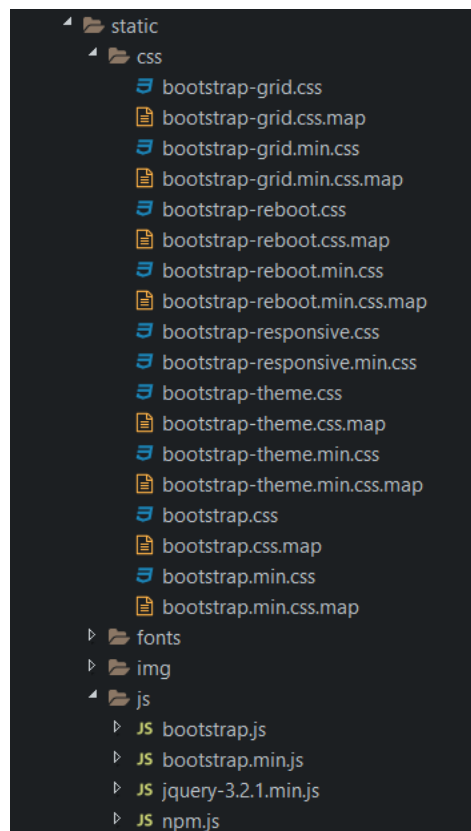


## Pengembangan

Saya memulai pengerjaan *project* ini dengan pertama memahami struktur dari *database* SIDUK yang telah diberikan. Setelah mendapat gambaran mengenai struktur *database* tersebut, saya membuat *model-model* yang bersangkutan. *Model* yang saya buat terdapat lima buah, masing-masing merepresentasikan tiap-tiap *entity* dari *database*. Setelah membuat *model*, saya melanjutkan pengerjaan *project* dengan membuat *file-file controller*, *mapper*, dan *service*. Dalam *file-file* tersebut, saya membuat *method-method basic* seperti *select*, *update*, *delete*, dan lainnya. Pengerjaan *project* selanjutnya dilakukan berdasarkan fitur-fitur yang akan dibuat.

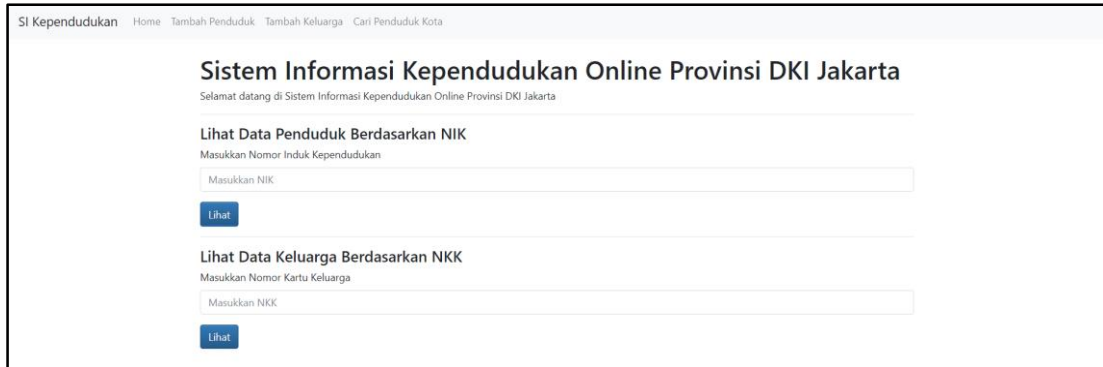
Selain tahap-tahap diatas, saya juga meng-*import* sejumlah *resources* bersifat *static* seperti *Bootstrap* dan *jQuery* dengan struktur seperti berikut:



Selain *static resources* yang letakkan langsung pada *directory resources > static*, saya juga menggunakan sejumlah CDN *online* seperti CDN untuk *DataTable*. Sehingga, koneksi internet dibutuhkan dalam menjalankan aplikasi ini.

## 1. Tampilkan Data Penduduk Berdasarkan NIK

Dalam pengerjaan fitur ini, saya memulai dengan membuat *view index* dan *navbar* yang menyerupai contoh yang telah diberikan sebagai berikut:



Pada *page* ini, terdapat *form* untuk melakukan pencarian penduduk berdasarkan NIK dan keluarga berdasarkan nomor KK dengan metode *GET*. Setelah itu, untuk mengetesnya, saya membuat *controller* dengan nama *IndexController* yang menampilkan *view index.html*.

```
package com.example.controller;

import org.springframework.stereotype.Controller;

@Controller
public class IndexController {

    @RequestMapping("/")
    public String index() {
        return "index";
    }
}
```

Pada *package* DAO, saya membuat *method-method* pada *mapper-mapper* yang berhubungan dengan fitur ini, yaitu semua *mapper* yang ada dalam *project*. *Method-method* yang saya buat digunakan untuk me-*retrieve* data atau *object* yang dibutuhkan dalam menampilkan data.

Pada *PendudukMapper* saya membuat *method* yang mengembalikan sebuah *object* berupa *PendudukModel* berdasarkan NIK penduduk tersebut. Kemudian pada *PendudukService*, saya membuat *method-method* bersangkutan yang juga memanggil

*method* pada *PendudukMapper*. Berikut *method-method* yang saya buat secara berurutan pada *PendudukMapper*, *PendudukService*, dan *PendudukServiceDatabase*:

```
@Select("SELECT * FROM penduduk WHERE nik = #{nik}")
@Results(value = {
    @Result(property="idPenduduk", column="id"),
    @Result(property="nik", column="nik"),
    @Result(property="namaPenduduk", column="nama"),
    @Result(property="tempatLahir", column="tempat_lahir"),
    @Result(property="tanggalLahir", column="tanggal_lahir"),
    @Result(property="jenisKelamin", column="jenis_kelamin"),
    @Result(property="isWni", column="is_wni"),
    @Result(property="idKeluarga", column="id_keluarga"),
    @Result(property="agama", column="agama"),
    @Result(property="pekerjaan", column="pekerjaan"),
    @Result(property="statusPerkawinan", column="status_perkawinan"),
    @Result(property="statusDalamKeluarga", column="status_dalam_keluarga"),
    @Result(property="golonganDarah", column="golongan_darah"),
    @Result(property="isWafat", column="is_wafat")
})
PendudukModel selectPenduduk(@Param("nik") String nik);
```

```
PendudukModel selectPenduduk(String nik);
```

```
@Override
public PendudukModel selectPenduduk(String nik) {
    return pendudukMapper.selectPenduduk(nik);
}
```

Pada *KeluargaMapper* saya membuat *method* yang mengembalikan sebuah *object* berupa *KeluargaModel* berdasarkan ID Keluarga yang terdapat pada *PendudukModel* penduduk sebelumnya. Kemudian pada *KeluargaService*, saya membuat *method-method* bersangkutan yang juga memanggil *method* pada *KeluargaMapper*. Berikut *method-method* yang saya buat secara berurutan pada *KeluargaMapper*, *KeluargaService*, dan *KeluargaServiceDatabase*:

```
@Select("SELECT * FROM keluarga WHERE id = #{idKeluarga}")
@Results(value = {
    @Result(property="idKeluarga", column="id"),
    @Result(property="nkk", column="nomor_kk"),
    @Result(property="alamat", column="alamat"),
    @Result(property="rt", column="RT"),
    @Result(property="rw", column="RW"),
    @Result(property="idKelurahan", column="id_kelurahan"),
    @Result(property="isTidakBerlaku", column="is_tidak_berlaku")
})
KeluargaModel selectKeluargaId(@Param("idKeluarga") int idKeluarga);
```

```
KeluargaModel selectKeluargaId(int idKeluarga);
```

```
@Override  
public KeluargaModel selectKeluargaId(int idKeluarga) {  
    return keluargaMapper.selectKeluargaId(idKeluarga);  
}
```

Pada *KelurahanMapper* saya membuat *method* yang mengembalikan sebuah *object* berupa *KelurahanModel* berdasarkan ID Kelurahan yang terdapat pada *KeluargaModel* Keluarga sebelumnya. Kemudian pada *KelurahanService*, saya membuat *method-method* bersangkutan yang juga memanggil *method* pada *KelurahanMapper*. Berikut *method-method* yang saya buat secara berurutan pada *KelurahanMapper*, *KelurahanService*, dan *KelurahanServiceDatabase*:

```
@Select("SELECT * FROM kelurahan WHERE id = #{idKelurahan}")  
@Results(value = {  
    @Result(property="idKelurahan", column="id"),  
    @Result(property="idKecamatan", column="id_kecamatan"),  
    @Result(property="kodeKelurahan", column="kode_kelurahan"),  
    @Result(property="namaKelurahan", column="nama_kelurahan"),  
    @Result(property="kodePos", column="kode_pos"),  
    @Result(property="listKeluarga", column="id",  
        javaType = List.class,  
        many=@Many(select="selectListKeluarga"))*/  
}))  
KelurahanModel selectKelurahanId(@Param("idKelurahan") int idKelurahan);
```

```
KelurahanModel selectKelurahanId(int idKelurahan);
```

```
@Override  
public KelurahanModel selectKelurahanId(int idKelurahan) {  
    return kelurahanMapper.selectKelurahanId(idKelurahan);  
}
```

Pada *KecamatanMapper* saya membuat *method* yang mengembalikan sebuah *object* berupa *KecamatanModel* berdasarkan ID Kecamatan yang terdapat pada *KelurahanModel* Kelurahan sebelumnya. Kemudian pada *KecamatanService*, saya membuat *method-method* bersangkutan yang juga memanggil *method* pada *KecamatanMapper*. Berikut *method-method* yang saya buat secara berurutan pada *KecamatanMapper*, *KecamatanService*, dan *KecamatanServiceDatabase*:

```
@Select("SELECT * FROM kecamatan WHERE id = #{idKecamatan}")
@Results(value = {
    @Result(property="idKecamatan", column="id"),
    @Result(property="idKota", column="id_kota"),
    @Result(property="kodeKecamatan", column="kode_kecamatan"),
    @Result(property="namaKecamatan", column="nama_kecamatan")
})
KecamatanModel selectKecamatanId(@Param("idKecamatan") int idKecamatan);
```

```
KecamatanModel selectKecamatanId(@Param("idKecamatan") int idKecamatan);
```

```
@Override
public KecamatanModel selectKecamatanId(@Param("idKecamatan") int idKecamatan) {
    return kecamatanMapper.selectKecamatanId(idKecamatan);
}
```

Pada *KotaMapper* saya membuat *method* yang mengembalikan sebuah *object* berupa *KotaModel* berdasarkan ID Kota yang terdapat pada *KecamatanModel* Kecamatan sebelumnya. Kemudian pada *KotaService*, saya membuat *method-method* bersangkutan yang juga memanggil *method* pada *KotaMapper*. Berikut *method-method* yang saya buat secara berurutan pada *KotaMapper*, *KotaService*, dan *KotaServiceDatabase*:

```
@Select("SELECT * FROM kota WHERE id = #{idKota}")
@Results(value = {
    @Result(property="idKota", column="id"),
    @Result(property="kodeKota", column="kode_kota"),
    @Result(property="namaKota", column="nama_kota"),
})
KotaModel selectKotaId(@Param("idKota") int idKota);
```

```
KotaModel selectKotaId(@Param("idKota") int idKota);
```

```
@Override
public KotaModel selectKotaId(@Param("idKota") int idKota) {
    return kotaMapper.selectKotaId(idKota);
}
```

Setelah selesai membuat semua *method-method* tersebut, saya membuat *method viewPenduduk* pada *PendudukController* sebagai berikut:

```
@RequestMapping(value = "/penduduk", method = RequestMethod.GET)
public String viewPenduduk(Model model, @RequestParam(value = "nik", required = false) String nik) {
    PendudukModel penduduk = pendudukDAO.selectPenduduk(nik);

    if (penduduk != null) {
        KeluargaModel keluarga = keluargaDAO.selectKeluargaId(penduduk.getIdKeluarga());
        KelurahanModel kelurahan = kelurahanDAO.selectKelurahanId(keluarga.getIdKelurahan());
        KecamatanModel kecamatan = kecamatanDAO.selectKecamatanId(kelurahan.getIdKecamatan());
        KotaModel kota = kotaDAO.selectKotaId(kecamatan.getIdKota());

        model.addAttribute("penduduk", penduduk);
        model.addAttribute("keluarga", keluarga);
        model.addAttribute("kelurahan", kelurahan);
        model.addAttribute("kecamatan", kecamatan);
        model.addAttribute("kota", kota);
        return "view-penduduk";
    } else {
        model.addAttribute("nik", nik);
        return "error-penduduk";
    }
}
```

Method ini menerima parameter berupa NIK yang berasal dari form pada *index.html* dengan *RequestMethod* berupa GET. Kemudian dicari penduduk dengan NIK tersebut yang kemudian di-assign ke variabel penduduk. Lalu dilakukan pengecekan ada tidaknya penduduk dengan NIK tersebut dengan membandingkan penduduk tersebut dengan *null*. Apabila *null*, berarti penduduk yang dicari tidak ada, sehingga akan ditampilkan *error page*. Apabila tidak *null*, data akan diproses lebih lanjut. Object keluarga, kelurahan, kecamatan, dan kota juga di-retrieve berdasarkan data penduduk dan data-data lain yang bersangkutan. Kemudian object-object tersebut akan diletakkan pada *model* untuk ditampilkan pada *view*. Kemudian *method* akan me-return *view-penduduk* untuk menampilkan *view* sebagai berikut:

SI Kependudukan Home Tambah Penduduk Tambah Keluarga Cari Penduduk Kota

Lihat Data Penduduk - 3101010302150003

Ubah Data

NIK	3101010302150003
Nama	Kairav Rekso Hidayanto M.Kom.
Tempat/Tanggal Lahir	Jakarta, 2015-02-03
Alamat	Jr. Kartini No. 626
Jenis Kelamin	Laki-laki
RT/RW	179/097
Kelurahan/Desa	PULAU TIDUNG
Kecamatan	KEPULAUAN SERIBU SELATAN
Kota	KABUPATEN KEPULAUAN SERIBU
Golongan Darah	O-
Agama	Katholik
Status Perkawinan	Belum Kawin
Status dalam Keluarga	Anak
Pekerjaan	BELUM/TIDAK BEKERJA
Kewarganegaraan	WNI
ID Keluarga	7 <a href="#">Lihat Keluarga</a>
Status	Hidup <a href="#">Set Wafat</a>

## 2. Tampilkan Data Keluarga Beserta Daftar Anggotanya Berdasarkan Nomor

Cara kerja fitur ini hampir sama dengan fitur pertama. Saya memberikan tambahan pada *PendudukMapper* sebuah *method* yang mengambil *list* penduduk berdasarkan ID Keluarga mereka.

```
@Select("SELECT * FROM penduduk WHERE penduduk.id_keluarga = #{idKeluarga}")
@Results(value = {
    @Result(property="idPenduduk", column="id"),
    @Result(property="nik", column="nik"),
    @Result(property="namaPenduduk", column="nama"),
    @Result(property="tempatLahir", column="tempat_lahir"),
    @Result(property="tanggalLahir", column="tanggal_lahir"),
    @Result(property="jenisKelamin", column="jenis_kelamin"),
    @Result(property="isWni", column="is_wni"),
    @Result(property="idKeluarga", column="id_keluarga"),
    @Result(property="agama", column="agama"),
    @Result(property="pekerjaan", column="pekerjaan"),
    @Result(property="statusPerkawinan", column="status_perkawinan"),
    @Result(property="statusDalamKeluarga", column="status_dalam_keluarga"),
    @Result(property="golonganDarah", column="golongan_darah"),
    @Result(property="isWafat", column="is_wafat")
})
List<PendudukModel> selectListPendudukByKeluarga(@Param("idKeluarga") int idKeluarga);
```

Pada *KeluargaController* saya membuat *method* sebagai berikut:

```
@RequestMapping(value = "/keluarga", method = RequestMethod.GET)
public String viewKeluarga(Model model, @RequestParam(value = "nkk", required = false) String nkk) {
    KeluargaModel keluarga = keluargaDAO.selectKeluarga(nkk);

    if(keluarga != null) {
        KelurahanModel kelurahan = kelurahanDAO.selectKelurahanId(keluarga.getIdKelurahan());
        KecamatanModel kecamatan = kecamatanDAO.selectKecamatanId(kelurahan.getIdKecamatan());
        KotaModel kota = kotaDAO.selectKotaId(kecamatan.getIdKota());

        model.addAttribute("listPenduduk", pendudukDAO.selectListPendudukByKeluarga(keluarga.getIdKeluarga()));
        model.addAttribute("keluarga", keluarga);
        model.addAttribute("kelurahan", kelurahan);
        model.addAttribute("kecamatan", kecamatan);
        model.addAttribute("kota", kota);
        return "view-keluarga";
    } else {
        model.addAttribute("nkk", nkk);
        return "error-keluarga";
    }
}
```

Kerjanya hampir sama dengan *method* pada *PendudukController*, perbedaan terdapat pada pemeran utamanya yaitu keluarga dan juga ditambahkan *listPenduduk* yang nantinya ditampilkan pada *view*.

### 3. Menambahkan Penduduk Baru Sebagai Anggota Keluarga

Pada *PendudukMapper*, saya membuat *method* untuk menambahkan penduduk ke *database* sebagai berikut:

```
@Insert("INSERT INTO penduduk (nik, nama, tempat_lahir, tanggal_lahir, jenis_kelamin, "
+ "is_wni, id_keluarga, agama, pekerjaan, status_perkawinan, status_dalam_keluarga, "
+ "golongan_darah, is_wafat) VALUES (#{nik}, #{namaPenduduk}, #{tempatLahir}, "
+ " #{tanggalLahir}, #{jenisKelamin}, #{isWni}, #{idKeluarga}, "
+ " #{agama}, #{pekerjaan}, #{statusPerkawinan}, #{statusDalamKeluarga}, "
+ " #{golonganDarah}, #{isWafat})")
void addPenduduk(PendudukModel penduduk);
```

Selain itu, pada *service* saya juga membuat *method* dengan nama sama yang berguna sebagai implementasi yang akan memanggil *method* pada *PendudukMapper*.

Pada *PendudukController* saya membuat *method* sebagai berikut:

```
@RequestMapping(value = "/penduduk/tambah")
public String addPenduduk(@ModelAttribute("penduduk") PendudukModel penduduk) {
    return "add-penduduk";
}

@RequestMapping(value = "/penduduk/tambah", method = RequestMethod.POST)
public String addPendudukSubmit(Model model, @ModelAttribute("penduduk") PendudukModel penduduk, BindingResult result) {
    if(result.hasErrors()) {
        return "error-page";
    } else {
        KeluargaModel keluarga = keluargaDAO.selectKeluargaId(penduduk.getIdKeluarga());
        KelurahanModel kelurahan = kelurahanDAO.selectKelurahanId(keluarga.getIdKelurahan());
        KecamatanModel kecamatan = kecamatanDAO.selectKecamatanId(kelurahan.getIdKecamatan());

        model.addAttribute("nik", penduduk.resetNik(kecamatan, pendudukDAO));
        pendudukDAO.addPenduduk(penduduk);
        return "success-penduduk";
    }
}
```

*Method* pertama digunakan untuk menangkap *url* dan me-redirect ke view *add-penduduk* yang berupa *form*, dan *method* kedua merupakan *method* untuk memproses hasil *form* dan menambahkannya ke *database* dengan memanggil *method-method* pada DAO.

#### 4. Menambahkan Keluarga Baru

Kerjanya hampir sama dengan fitur menambah penduduk. Pertama saya membuat *method* pada *mapper* sebagai berikut:

```
@Insert("INSERT INTO keluarga (nomor_kk, alamat, rt, rw, id_kelurahan) "
        + "VALUES ({nk}, #{alamat}, #{rt}, #{rw}, #{idKelurahan})")
void addKeluarga(KeluargaModel keluarga);
```

Selain itu, pada *service* saya juga membuat *method* dengan nama sama yang berguna sebagai implementasi yang akan memanggil *method* pada *KeluargaMapper*.

Pada *KeluargaController*, saya membuat *method* sebagai berikut:



```
@RequestMapping(value = "/keluarga/tambah")
public String addKeluarga(@ModelAttribute("keluarga") KeluargaModel keluarga) {
    return "add-keluarga";
}

@RequestMapping(value = "/keluarga/tambah", method = RequestMethod.POST)
public String addKeluargaSubmit(Model model, @ModelAttribute("keluarga") KeluargaModel keluarga, BindingResult result) {
    if(result.hasErrors()) {
        return "error-page";
    } else {
        KelurahanModel kelurahan = kelurahanDAO.selectKelurahanId(keluarga.getIdKelurahan());
        KecamatanModel kecamatan = kecamatanDAO.selectKecamatanId(kelurahan.getIdKecamatan());

        model.addAttribute("nkk", keluarga.resetNkk(kecamatan, pendudukDAO, keluargaDAO));
        keluargaDAO.addKeluarga(keluarga);
        return "success-keluarga";
    }
}
```

Sama seperti sebelumnya, *method* pertama digunakan sebagai penangkap *url* ang *redirect* ke view *add-keluarga* untuk mengisi *form*, *method* kedua digunakan untuk memproses data dari *form* pada *method* pertama. Kemudian hasilnya akan dimasukkan ke *database*.

## 5. Mengubah Data Penduduk

Pertama, saya *membuat mapper Update* untuk melakukan *update* terhadap *database*. *Mapper* pada *PendudukMapper* yang saya buat adalah sebagai berikut:

```
@Update("UPDATE penduduk SET nik=#{nik}, nama=#{namaPenduduk}, "
+ "tempat_lahir=#{tempatLahir}, tanggal_lahir=#{tanggalLahir}, "
+ "jenis_kelamin=#{jenisKelamin}, is_wni=#{isWni}, "
+ "id_keluarga=#{idKeluarga}, agama=#{agama}, pekerjaan=#{pekerjaan}, "
+ "status_perkawinan=#{statusPerkawinan}, "
+ "status_dalam_keluarga=#{statusDalamKeluarga}, "
+ "golongan_darah=#{golonganDarah}, is_wafat=#{isWafat} "
+ "WHERE id=#{idPenduduk}")
void updatePenduduk(PendudukModel penduduk);
```

Selain itu, pada *service* saya juga membuat *method* dengan nama sama yang berguna sebagai implementasi yang akan memanggil *method* pada *PendudukMapper*.

Pada *PendudukController*, saya membuat *method-method* berikut:

```
@RequestMapping(value = "/penduduk/ubah/{nik}")
public String updatePenduduk(@PathVariable(value="nik") String nik, Model model) {
    PendudukModel penduduk = pendudukDAO.selectPenduduk(nik);

    if(penduduk != null) {
        model.addAttribute("penduduk", penduduk);
        return "update-penduduk";
    } else {
        model.addAttribute("nik", nik);
        return "error-penduduk";
    }
}

@RequestMapping(value = "/penduduk/ubah/{nik}", method = RequestMethod.POST)
public String updatePendudukSubmit(Model model, @ModelAttribute("penduduk") PendudukModel penduduk, BindingResult result) {
    if(result.hasErrors()) {
        return "error-page";
    } else {
        KeluargaModel keluarga = keluargaDAO.selectKeluargaId(penduduk.getIdKeluarga());
        KelurahanModel kelurahan = kelurahanDAO.selectKelurahanId(keluarga.getIdKelurahan());
        KecamatanModel kecamatan = kecamatanDAO.selectKecamatanId(kelurahan.getIdKecamatan());

        model.addAttribute("nik", penduduk.getNik());
        penduduk.resetNik(kecamatan, pendudukDAO);
        model.addAttribute("nikBaru", penduduk.getNik());

        pendudukDAO.updatePenduduk(penduduk);

        keluarga.checkBerlaku(pendudukDAO, keluargaDAO);
        keluargaDAO.setBerlaku(keluarga.getIsTidakBerlaku(), keluarga.getIdKeluarga());

        return "success-penduduk-edit";
    }
}
```

Method pertama berguna untuk mengambil *object* dan memasukkannya ke *model* untuk digunakan datanya pada *form* pada *update-penduduk* dan kemudian *redirect* ke *view update-penduduk*. Method kedua berguna untuk menerima dan memproses *data* dari *form*, kemudian memanggil *method-method* DAO untuk melakukan perubahan pada *database*.

## 6. Mengubah Data Keluarga

Pertama, saya membuat *mapper Update* untuk melakukan *update* terhadap *database*. *Mapper* pada *KeluargaMapper* yang saya buat adalah sebagai berikut:

```
@Update("UPDATE keluarga SET nomor_kk=#{nkk}, alamat=#{alamat}, "
+ "RT=#{rt}, RW=#{rw},id_kelurahan=#{idKelurahan}, "
+ "is_tidak_berlaku=#{isTidakBerlaku} "
+ "WHERE id=#{idKeluarga}")
void updateKeluarga(KeluargaModel keluarga);
```

Selain itu, pada *service* saya juga membuat *method* dengan nama sama yang berguna sebagai implementasi yang akan memanggil *method* pada *KeluargaMapper*.

Pada *KeluargaController*, saya membuat *method-method* berikut:

```
@RequestMapping(value = "/keluarga/ubah/{nkk}")
public String updateKeluarga(Model model, @PathVariable(value = "nkk") String nkk) {
    KeluargaModel keluarga = keluargaDAO.selectKeluarga(nkk);

    if(keluarga != null) {
        model.addAttribute("keluarga", keluarga);
        return "update-keluarga";
    } else {
        model.addAttribute("nkk", nkk);
        return "error-keluarga";
    }
}

@RequestMapping(value = "/keluarga/ubah/{nkk}", method = RequestMethod.POST)
public String updateKeluargaSubmit(Model model, @ModelAttribute("keluarga") KeluargaModel keluarga, BindingResult result) {
    if(result.hasErrors()) {
        return "error-page";
    } else {
        KelurahanModel kelurahan = kelurahanDAO.selectKelurahanId(keluarga.getIdKelurahan());
        KecamatanModel kecamatan = kecamatanDAO.selectKecamatanId(keluurahan.getIdKecamatan());

        model.addAttribute("nkk", keluarga.getNkk());
        keluarga.resetNkk(kecamatan, pendudukDAO, keluargaDAO);
        model.addAttribute("nkkBaru", keluarga.getNkk());
        keluargaDAO.updateKeluarga(keluarga);

        return "success-keluarga-edit";
    }
}
```

*Method* pertama berguna untuk mengambil *object* dan memasukkannya ke *model* untuk digunakan datanya pada *form* pada *update-keluarga* dan kemudian me-redirect ke *view update-penduduk*. *Method* kedua berguna untuk menerima dan memproses *data* dari *form*, kemudian memanggil *method-method* DAO untuk melakukan perubahan pada *database*.

## 7. Mengubah Status Kematian Penduduk

Pada *PendudukMapper* dan *KeluargaMapper* saya membuat *method-method* untuk mengubah status kematian dan keberlakuan KK pada *database* sebagai berikut:

```
@Update("UPDATE penduduk SET is_wafat=#{isWafat} WHERE id=#{idPenduduk}")
void wafatPenduduk(@Param("isWafat") int isWafat, @Param("idPenduduk") int idPenduduk);
```

```
@Update("UPDATE keluarga SET is_tidak_berlaku=#{isTidakBerlaku} WHERE id=#{idKeluarga}")
void setBerlaku(@Param("isTidakBerlaku") int isTidakBerlaku, @Param("idKeluarga") int idKeluarga);
```

Selain itu, saya juga membuat *method-method service* yang bersangkutan agar dapat digunakan sebagai DAO.

Pada *PendudukController*, saya membuat *method* sebagai berikut:

```
@RequestMapping(value = "/penduduk/mati", method = RequestMethod.POST)
public String wafatPendudukSubmit(Model model,
    @RequestParam(value = "isWafat", required = false) Integer isWafat,
    @RequestParam(value = "idPenduduk", required = false) Integer idPenduduk,
    @RequestParam(value = "idKeluarga", required = false) Integer idKeluarga,
    @RequestParam(value = "nik", required = false) String nik) {

    if(isWafat != null && idPenduduk != null) {
        pendudukDAO.wafatPenduduk(isWafat, idPenduduk);

        KeluargaModel keluarga = keluargaDAO.selectKeluargaId(idKeluarga.intValue());
        keluarga.checkBerlaku(pendudukDAO, keluargaDAO);
        keluargaDAO.setBerlaku(keluarga.getIsTidakBerlaku(), idKeluarga);

        model.addAttribute("nik", nik);
        model.addAttribute("isWafat", isWafat);
        return "success-mati";
    } else {
        return "error-page";
    }
}
```

*Method* ini menerima status kematian penduduk yang baru, me-set status kematian baru, lalu mengecek dan mengubah keberlakuan KK dari penduduk berdasarkan status kematian para penduduk.

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

Saya membaut *method* pada *PendudukController* sebagai berikut:

```
@RequestMapping(value = "/penduduk/cari", method = RequestMethod.GET)
public String searchPendudukKota(Model model,
    @RequestParam(value = "kt", required = false) Integer idKota,
    @RequestParam(value = "kc", required = false) Integer idKecamatan,
    @RequestParam(value = "kl", required = false) Integer idKelurahan) {

    if(idKota != null && idKecamatan != null && idKelurahan != null) {
        model.addAttribute("kelurahan", kelurahanDAO.selectKelurahanId(idKelurahan));
        List<PendudukModel> listPenduduk = pendudukDAO.selectListPendudukByKelurahan(idKelurahan);
        model.addAttribute("listPenduduk", listPenduduk);

        Collections.sort(listPenduduk);
        PendudukModel pendudukTermuda = listPenduduk.get(listPenduduk.size() - 1);
        PendudukModel pendudukTertua = listPenduduk.get(0);

        model.addAttribute("pendudukTermuda", pendudukTermuda);
        model.addAttribute("pendudukTertua", pendudukTertua);

        return "search-penduduk-table";
    } else if (idKota != null && idKecamatan != null) {
        model.addAttribute("kota", kotaDAO.selectKotaId(idKota));
        model.addAttribute("kecamatan", kecamatanDAO.selectKecamatanId(idKecamatan));
        model.addAttribute("listKelurahan", kelurahanDAO.selectListKelurahanByKecamatan(idKecamatan));

        return "search-penduduk-kelurahan";
    } else if (idKota != null) {
        model.addAttribute("kota", kotaDAO.selectKotaId(idKota));
        model.addAttribute("listKecamatan", kecamatanDAO.selectListKecamatanByKota(idKota));

        return "search-penduduk-kecamatan";
    } else {
        model.addAttribute("listKota", kotaDAO.selectKotaAll());
        return "search-penduduk-kota";
    }
}
```

*Method* ini adalah *method* utama pada fitur ini. *Method* ini akan melakukan pengecekan terhadap nilai dari ID-ID hasil *request* GET dan memberikan *view* sesuai nilai-nilai tersebut. Kemudian, setelah data-data sudah ada, untuk menampilkan penduduk pada kelurahan tertentu saya membuat *method* pada *PendudukMapper* untuk memilih penduduk berdasarkan kelurahan untuk mengoptimasi *query*.

```
@Select("SELECT penduduk.* FROM penduduk WHERE penduduk.id_keluarga IN (SELECT keluarga.id FROM keluarga WHERE keluarga.id_kelurahan = #{idKelurahan})")
@Results(value = {
    @Result(property="idPenduduk", column="id"),
    @Result(property="nik", column="nik"),
    @Result(property="namaPenduduk", column="nama"),
    @Result(property="tempatLahir", column="tempat_lahir"),
    @Result(property="tanggalLahir", column="tanggal_lahir"),
    @Result(property="jenisKelamin", column="jenis_kelamin"),
    @Result(property="isWni", column="is_wni"),
    @Result(property="idKeluarga", column="id_keluarga"),
    @Result(property="agama", column="agama"),
    @Result(property="pekerjaan", column="pekerjaan"),
    @Result(property="statusPerkawinan", column="status_perkawinan"),
    @Result(property="statusDalamKeluarga", column="status_dalam_keluarga"),
    @Result(property="golonganDarah", column="golongan_darah"),
    @Result(property="isWafat", column="is_wafat")
})
List<PendudukModel> selectListPendudukByKelurahan(@Param("idKelurahan") int idKelurahan);
```

## 9. [BONUS] Menampilkan Penduduk Paling Muda dan Paling Tua di Suatu Kelurahan

Untuk fitur ini, saya mengimplementasikan *Comparable* pada *PendudukModel* sebagai berikut:

```
@Override
public int compareTo(PendudukModel otherPenduduk) {

    DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
    try {
        Date thisDate = format.parse(getTanggalLahir());
        Date otherDate = format.parse(otherPenduduk.getTanggalLahir());
        return thisDate.compareTo(otherDate);
    } catch (Exception e) {

    }
    return 0;
}
```

Atribut yang dibandingkan adalah tanggal lahir dari penduduk, diurut dari tertua ke termuda.

Kemudian pada *method* *searchPendudukKota* pada *PendudukController* saya mengimplementasikannya sebagai berikut:

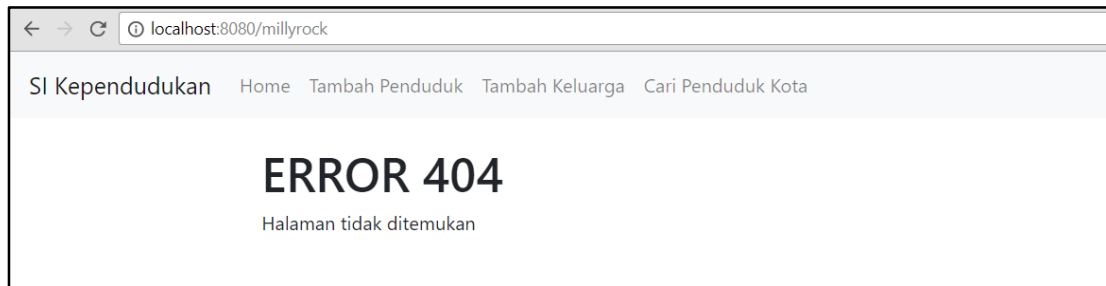
```
Collections.sort(listPenduduk);
PendudukModel pendudukTermuda = listPenduduk.get(listPenduduk.size() - 1);
PendudukModel pendudukTertua = listPenduduk.get(0);

model.addAttribute("pendudukTermuda", pendudukTermuda);
model.addAttribute("pendudukTertua", pendudukTertua);
```

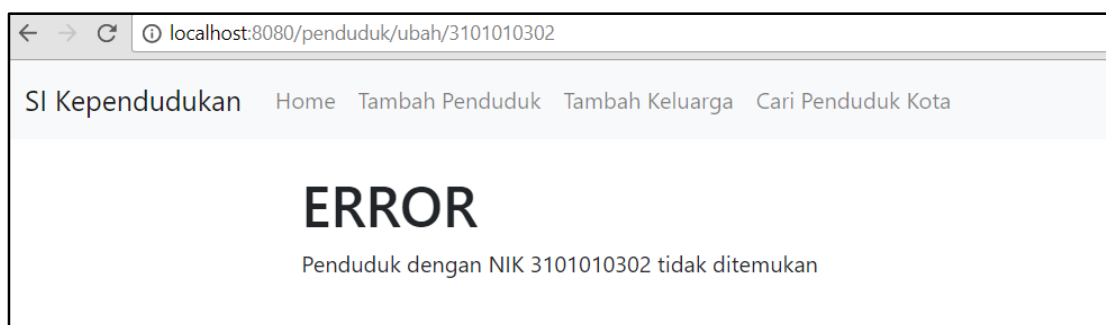
Pertama *sort List* yang berisi penduduk dalam suatu kelurahan, kemudian ambil *index* pertama untuk yang tertua, dan *index* terakhir untuk termuda. Kemudian dimasukkan ke dalam *model*.

## 10. [BONUS] Menambahkan *Error Page*

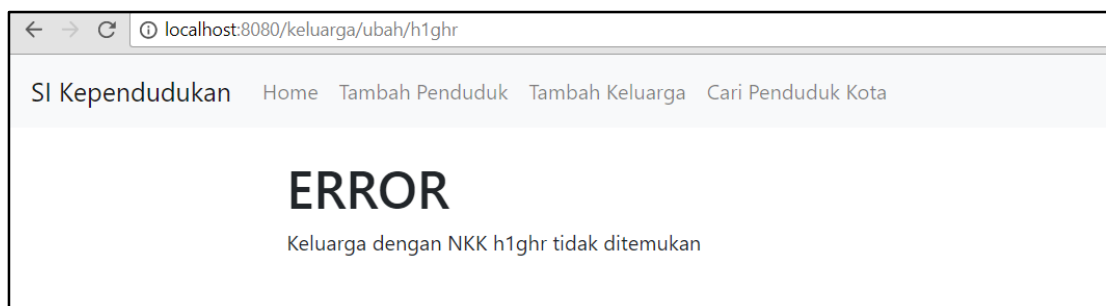
*Error 404* (Halaman tidak ditemukan)



*Error penduduk tidak ditemukan*



*Error keluarga tidak ditemukan*



## 11. [BONUS] Fitur-Fitur Lain yang Mendukung Aplikasi Fitur Terkait

Fitur mengaktifkan kembali orang yang sudah di set nonaktif pada *view-penduduk*, berguna untuk memudahkan *reversal* apabila melakukan kesalahan atau hal lainnya.

Status	Wafat	Set Hidup
--------	-------	-----------

Fitur untuk melihat keluarga dari penduduk yang sedang dilihat datanya dengan menekan tombol pada *view-penduduk*. Memudahkan pengguna untuk bernavigasi dalam aplikasi.

ID Keluarga	1	Lihat Keluarga
-------------	---	----------------

Fitur mengubah data pada *view-penduduk*. Memudahkan pengguna agar tidak harus mengisi *url* untuk mengubah *data*.

### Lihat Data Penduduk - 3101010303890001

Ubah Data

NIK	3101010303890001
-----	------------------

Fitur mengubah data pada *view-penduduk*. Memudahkan pengguna agar tidak harus mengisi *url* untuk mengubah *data*.

### Lihat Data Keluarga - 3101010101050001

Ubah Data

NKK	3101010101050001
-----	------------------

Fitur untuk melihat *page* penduduk dengan menekan tombol pada *view-keluarga*. Berguna untuk memudahkan pengguna dalam bernavigasi dalam aplikasi.

	<b>Status</b>	
	Wafat	<a href="#">Lihat Penduduk</a>
	Hidup	<a href="#">Lihat Penduduk</a>

## Optimasi *Database*

Optimasi *database* yang saya lakukan adalah dengan membuat *indexing* terhadap *key* dan *foreign key* pada *database* untuk mempercepat proses *look-up*.

## Struktur *Project*

*Project* Tugas 1 ini terdiri atas 5 buah *package*. *Package-package* tersebut adalah sebagai berikut:



```
src/main/java
├── com.example
│   ├── ApapTugas01Application.java
│   ├── com.example.controller
│   │   ├── IndexController.java
│   │   │   ├── IndexController
│   │   ├── KeluargaController.java
│   │   │   ├── KeluargaController
│   │   ├── PendudukController.java
│   ├── com.example.dao
│   │   ├── KecamatanMapper.java
│   │   ├── KeluargaMapper.java
│   │   ├── KelurahanMapper.java
│   │   ├── KotaMapper.java
│   │   ├── PendudukMapper.java
│   │   │   ├── PendudukMapper
│   ├── com.example.model
│   │   ├── KecamatanModel.java
│   │   ├── KeluargaModel.java
│   │   ├── KelurahanModel.java
│   │   ├── KotaModel.java
│   │   ├── PendudukModel.java
│   │   │   ├── PendudukModel
│   ├── com.example.service
│   │   ├── KecamatanService.java
│   │   ├── KecamatanServiceDatabase.java
│   │   ├── KeluargaService.java
│   │   ├── KeluargaServiceDatabase.java
│   │   ├── KelurahanService.java
│   │   ├── KelurahanServiceDatabase.java
│   │   ├── KotaService.java
│   │   ├── KotaServiceDatabase.java
│   │   ├── PendudukService.java
│   │   │   ├── PendudukService
│   │   ├── PendudukServiceDatabase.java
```

- `com.example`

Pada *package* ini, terdapat *class* untuk menjalankan aplikasi.

- `com.example.controller`

Pada *package* ini, terdapat *controller-controller* dari *project*. *Controller* dibuat berdasarkan *endpoint*, yaitu penduduk dan keluarga, serta selain itu sebuah *controller* untuk *index*.

- `com.example.dao`

Pada *package* ini, terdapat *mapper-mapper* dari *project*. *Mapper* dibuat berdasarkan *model-model* dan *entity* pada *database* untuk memudahkan *maintenance*.

- `com.example.model`

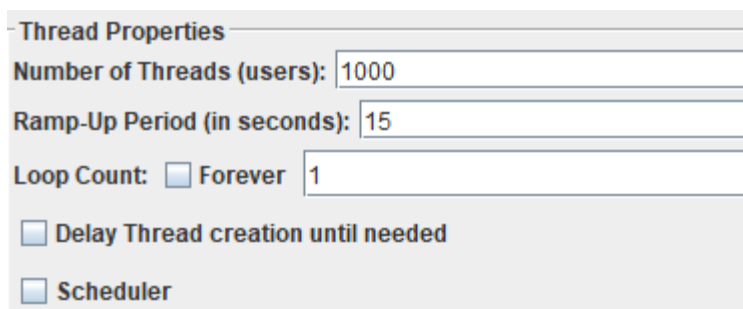
Pada *package* ini, terdapat *model-model* dari *project*. *Model* dibuat berdasarkan tabel-tabel atau *entity* pada *database* untuk memudahkan *maintenance*.

- `com.example.service`

Pada *package* ini, terdapat *service-service* dari *project*. *Service* dibuat berdasarkan *mapper*.

## Stress Testing

Berikut spesifikasi yang saya gunakan untuk *tress testing*



The image shows a 'Thread Properties' dialog box with the following settings:

- Number of Threads (users): 1000
- Ramp-Up Period (in seconds): 15
- Loop Count: ☐ Forever, 1
- ☐ Delay Thread creation until needed
- ☐ Scheduler

Fitur 1

Sample #	Start Time	Thread Name	Label ^	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
967	22:11:27.191	Thread Group 1-967	Penduduk by NIK	4	✓	3731	147	4	0
968	22:11:27.213	Thread Group 1-968	Penduduk by NIK	4	✓	3731	147	4	0
969	22:11:27.227	Thread Group 1-969	Penduduk by NIK	4	✓	3731	147	4	0
970	22:11:27.242	Thread Group 1-970	Penduduk by NIK	4	✓	3731	147	4	0
971	22:11:27.257	Thread Group 1-971	Penduduk by NIK	4	✓	3731	147	4	0
972	22:11:27.273	Thread Group 1-972	Penduduk by NIK	5	✓	3731	147	5	0
973	22:11:27.289	Thread Group 1-973	Penduduk by NIK	5	✓	3731	147	5	0
974	22:11:27.303	Thread Group 1-974	Penduduk by NIK	4	✓	3731	147	4	0
975	22:11:27.319	Thread Group 1-975	Penduduk by NIK	4	✓	3731	147	4	1
976	22:11:27.333	Thread Group 1-976	Penduduk by NIK	4	✓	3731	147	4	1
977	22:11:27.348	Thread Group 1-977	Penduduk by NIK	4	✓	3731	147	4	1
978	22:11:27.362	Thread Group 1-978	Penduduk by NIK	5	✓	3731	147	5	1
979	22:11:27.377	Thread Group 1-979	Penduduk by NIK	5	✓	3731	147	5	1
980	22:11:27.392	Thread Group 1-980	Penduduk by NIK	5	✓	3731	147	5	1
981	22:11:27.408	Thread Group 1-981	Penduduk by NIK	5	✓	3731	147	5	1
982	22:11:27.423	Thread Group 1-982	Penduduk by NIK	5	✓	3731	147	5	1
983	22:11:27.438	Thread Group 1-983	Penduduk by NIK	4	✓	3731	147	4	0
984	22:11:27.453	Thread Group 1-984	Penduduk by NIK	5	✓	3731	147	5	1
985	22:11:27.469	Thread Group 1-985	Penduduk by NIK	5	✓	3731	147	5	1
986	22:11:27.483	Thread Group 1-986	Penduduk by NIK	5	✓	3731	147	5	1
987	22:11:27.499	Thread Group 1-987	Penduduk by NIK	5	✓	3731	147	5	1
988	22:11:27.513	Thread Group 1-988	Penduduk by NIK	5	✓	3731	147	5	1
989	22:11:27.528	Thread Group 1-989	Penduduk by NIK	5	✓	3731	147	5	1
990	22:11:27.544	Thread Group 1-990	Penduduk by NIK	4	✓	3731	147	4	0
991	22:11:27.559	Thread Group 1-991	Penduduk by NIK	4	✓	3731	147	4	0
992	22:11:27.574	Thread Group 1-992	Penduduk by NIK	4	✓	3731	147	4	0
993	22:11:27.589	Thread Group 1-993	Penduduk by NIK	4	✓	3731	147	4	0
994	22:11:27.604	Thread Group 1-994	Penduduk by NIK	4	✓	3731	147	4	0
995	22:11:27.619	Thread Group 1-995	Penduduk by NIK	4	✓	3731	147	4	0
996	22:11:27.634	Thread Group 1-996	Penduduk by NIK	4	✓	3731	147	4	0
997	22:11:27.650	Thread Group 1-997	Penduduk by NIK	4	✓	3731	147	4	0
998	22:11:27.664	Thread Group 1-998	Penduduk by NIK	4	✓	3731	147	4	0
999	22:11:27.679	Thread Group 1-999	Penduduk by NIK	4	✓	3731	147	4	0
1000	22:11:27.693	Thread Group 1-1000	Penduduk by NIK	4	✓	3731	147	4	1

☐ Scroll automatically? ☐ Child samples? No of Samples 1000 Latest Sample 4 Average 5 Deviation 8

Fitur 2

Sample #	Start Time	Thread Name	Label ^	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	22:25:48.660	Thread Group 1-11	Penduduk by NIK	62	✓	4555	147	62	0
2	22:25:48.645	Thread Group 1-10	Penduduk by NIK	77	✓	4555	147	77	0
3	22:25:48.524	Thread Group 1-2	Penduduk by NIK	198	✓	4555	147	198	1
4	22:25:48.704	Thread Group 1-14	Penduduk by NIK	18	✓	4555	147	18	1
5	22:25:48.554	Thread Group 1-4	Penduduk by NIK	168	✓	4555	147	168	1
6	22:25:48.614	Thread Group 1-8	Penduduk by NIK	108	✓	4555	147	108	1
7	22:25:48.539	Thread Group 1-3	Penduduk by NIK	183	✓	4555	147	183	1
8	22:25:48.584	Thread Group 1-6	Penduduk by NIK	140	✓	4555	147	140	1
9	22:25:48.569	Thread Group 1-5	Penduduk by NIK	157	✓	4555	147	157	1
10	22:25:48.509	Thread Group 1-1	Penduduk by NIK	217	✓	4555	147	217	1
11	22:25:48.630	Thread Group 1-9	Penduduk by NIK	97	✓	4555	147	97	0
12	22:25:48.675	Thread Group 1-12	Penduduk by NIK	52	✓	4555	147	52	0
13	22:25:48.600	Thread Group 1-7	Penduduk by NIK	130	✓	4555	147	130	0
14	22:25:48.690	Thread Group 1-13	Penduduk by NIK	40	✓	4555	147	40	0
15	22:25:48.722	Thread Group 1-15	Penduduk by NIK	9	✓	4555	147	9	1
16	22:25:48.736	Thread Group 1-16	Penduduk by NIK	5	✓	4555	147	5	0
17	22:25:48.751	Thread Group 1-17	Penduduk by NIK	5	✓	4555	147	5	1
18	22:25:48.766	Thread Group 1-18	Penduduk by NIK	5	✓	4555	147	5	1
19	22:25:48.781	Thread Group 1-19	Penduduk by NIK	5	✓	4555	147	5	1
20	22:25:48.796	Thread Group 1-20	Penduduk by NIK	5	✓	4555	147	5	1
21	22:25:48.810	Thread Group 1-21	Penduduk by NIK	6	✓	4555	147	6	1
22	22:25:48.825	Thread Group 1-22	Penduduk by NIK	6	✓	4555	147	6	1
23	22:25:48.840	Thread Group 1-23	Penduduk by NIK	5	✓	4555	147	5	1
24	22:25:48.856	Thread Group 1-24	Penduduk by NIK	6	✓	4555	147	6	1
25	22:25:48.871	Thread Group 1-25	Penduduk by NIK	20	✓	4555	147	20	1
26	22:25:48.887	Thread Group 1-26	Penduduk by NIK	5	✓	4555	147	5	1
27	22:25:48.902	Thread Group 1-27	Penduduk by NIK	5	✓	4555	147	5	1
28	22:25:48.917	Thread Group 1-28	Penduduk by NIK	5	✓	4555	147	5	1
29	22:25:48.932	Thread Group 1-29	Penduduk by NIK	5	✓	4555	147	5	1
30	22:25:48.947	Thread Group 1-30	Penduduk by NIK	5	✓	4555	147	5	1
31	22:25:48.961	Thread Group 1-31	Penduduk by NIK	5	✓	4555	147	5	1
32	22:25:48.976	Thread Group 1-32	Penduduk by NIK	6	✓	4555	147	6	1
33	22:25:48.991	Thread Group 1-33	Penduduk by NIK	5	✓	4555	147	5	1
34	22:25:49.006	Thread Group 1-34	Penduduk by NIK	5	✓	4555	147	5	1

☐ Scroll automatically? ☐ Child samples? No of Samples 1000 Latest Sample 5 Average 6 Deviation 15

## Fitur 8

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename

Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label ^	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	22:34:06.370	Thread Group 1-1	Carl Penduduk	692	✓	233504	149	455	1
2	22:34:06.386	Thread Group 1-2	Carl Penduduk	698	✓	233504	149	683	0
3	22:34:06.416	Thread Group 1-4	Carl Penduduk	871	✓	233504	149	657	0
4	22:34:06.431	Thread Group 1-5	Carl Penduduk	871	✓	233504	149	858	0
5	22:34:06.461	Thread Group 1-7	Carl Penduduk	845	✓	233504	149	829	0
6	22:34:06.536	Thread Group 1-12	Carl Penduduk	1546	✓	233504	149	1223	0
7	22:34:06.476	Thread Group 1-8	Carl Penduduk	1644	✓	233504	149	1112	0
8	22:34:06.506	Thread Group 1-10	Carl Penduduk	1817	✓	233504	149	1800	0
9	22:34:06.446	Thread Group 1-6	Carl Penduduk	1976	✓	233504	149	1955	0
10	22:34:06.401	Thread Group 1-3	Carl Penduduk	2865	✓	233504	149	2849	1
11	22:34:06.762	Thread Group 1-27	Carl Penduduk	2635	✓	233504	149	2509	1
12	22:34:06.642	Thread Group 1-19	Carl Penduduk	2901	✓	233504	149	1694	0
13	22:34:07.079	Thread Group 1-48	Carl Penduduk	2506	✓	233504	149	2488	1
14	22:34:06.596	Thread Group 1-16	Carl Penduduk	3056	✓	233504	149	3039	0
15	22:34:06.491	Thread Group 1-9	Carl Penduduk	3301	✓	233504	149	3284	0
16	22:34:06.521	Thread Group 1-11	Carl Penduduk	3411	✓	233504	149	3389	0
17	22:34:06.973	Thread Group 1-41	Carl Penduduk	3126	✓	233504	149	3109	1
18	22:34:06.928	Thread Group 1-38	Carl Penduduk	3232	✓	233504	149	3213	1
19	22:34:06.702	Thread Group 1-23	Carl Penduduk	3461	✓	233504	149	3443	0
20	22:34:06.581	Thread Group 1-15	Carl Penduduk	3592	✓	233504	149	3368	0
21	22:34:06.611	Thread Group 1-17	Carl Penduduk	3625	✓	233504	149	3602	0
22	22:34:06.657	Thread Group 1-20	Carl Penduduk	3581	✓	233504	149	3563	1
23	22:34:06.566	Thread Group 1-14	Carl Penduduk	3863	✓	233504	149	3807	0
24	22:34:06.777	Thread Group 1-28	Carl Penduduk	3688	✓	233504	149	3669	1
25	22:34:06.551	Thread Group 1-13	Carl Penduduk	3981	✓	233504	149	3964	1
26	22:34:06.717	Thread Group 1-24	Carl Penduduk	3824	✓	233504	149	3806	0
27	22:34:06.627	Thread Group 1-18	Carl Penduduk	4094	✓	233504	149	3613	1

☐ Scroll automatically? ☐ Child samples? No of Samples 1000 Latest Sample 72928 Average 45089 Deviation 20998

Berdasarkan hasil *stress testing*, fitur 1 dan 2 berjalan dengan sangat cepat & *reliable* walaupun dengan jumlah *thread* yang cukup banyak. Sedangkan fitur 8 memiliki *average* yang cukup tinggi, tetapi masih *reliable* karena tidak terjadi *crash* atau *error* sehingga informasi sampai dengan tepat.