

Writeup Tugas Individu APAP

Pada pengerajan tugas ini, saya mulai dari menentukan model serta atribut yang dibutuhkan untuk Sistem Kependudukan Online Provinsi Jakarta. Dari deskripsi soal yang ada, saya memutuskan untuk membuat 5 model yang akan merepresentasikan 5 tabel dari database yaitu PendudukModel, KeluargaModel, KelurahanModel, KecamatanModel, dan KotaModel. Setelah menentukan model yang akan saya buat. Untuk bagian dao, service , view, dan sebagainya akan mengikuti kebutuhan dari fitur-fitur yang saya buat. Selain itu tugas ini didukung oleh database yang telah tersedia yaitu siduk_dki.

Dalam tugas ini, saya mengerjakan 8 fitur. Fitir-fitur tersebut adalah:

- Tampilkan Data Penduduk Berdasarkan NIK
- Tampilkan Data Keluarga Berdasarkan NKK
- Menambahkan Penduduk Baru sebagai Anggota Keluarga
- Menambah Keluarga Baru
- Mengubah Data Penduduk
- Mengubah Data Keluarga
- Mengubah Status Kematian Penduduk
- Tampilkan Data Penduduk berdasarkan Kota/Kabupaten, Kecamatan, dan Kelurahan tertentu

Berikut penjelasan dari masing-masing fitur yang saya buat,

1. Fitur Tampilkan Data Penduduk Berdasarkan NIK

Selamat datang di Sistem Informasi Kependudukan Online Provinsi DKI Jakarta

Lihat Data Penduduk Berdasarkan NIK
Masukan Nomor Induk Kependudukan
Masukan nomor NIK

LIHAT >

Lihat Data Keluarga Berdasarkan NKK
Masukan Nomor KK
Masukan nomor NKK

LIHAT >

Fitur ini akan menampilkan data seorang penduduk jika user mengisi NIK penduduk tersebut dalam field yang tersedia. Ketika tombol Lihat di klik maka request akan dikirim dan mengembalikan halaman yang berisi data penduduk lengkap dengan alamat.

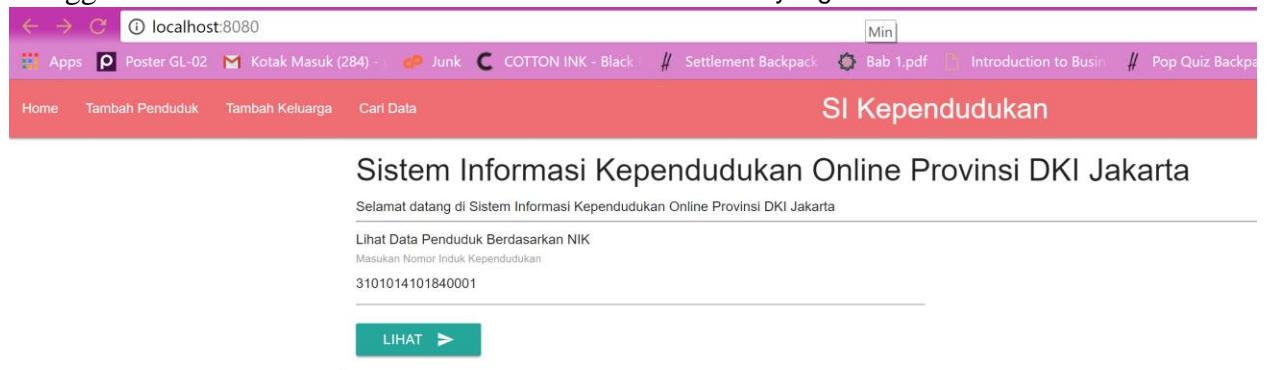
Disa Ainun Yolanna

1506689654

APAP A

Contoh scenario fitur:

Pengguna memasukan NIK : 3101014101840001 dalam field yang disediakan.



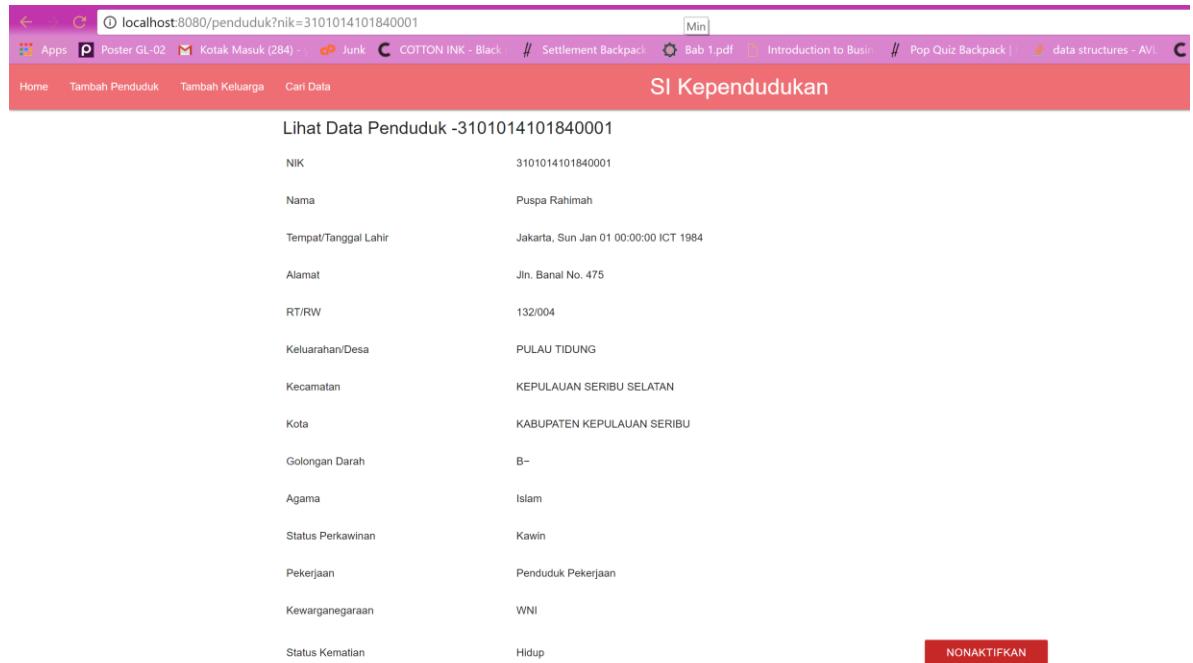
Sistem Informasi Kependudukan Online Provinsi DKI Jakarta

Selamat datang di Sistem Informasi Kependudukan Online Provinsi DKI Jakarta

Lihat Data Penduduk Berdasarkan NIK
Masukan Nomor Induk Kependudukan
3101014101840001

LIHAT >

Maka akan menampilkan



Lihat Data Penduduk -3101014101840001

NIK	3101014101840001
Nama	Puspa Rahimah
Tempat/Tanggal Lahir	Jakarta, Sun Jan 01 00:00:00 ICT 1984
Alamat	Jln. Banal No. 475
RT/RW	132/004
Keluarahan/Desa	PULAU TIDUNG
Kecamatan	KEPULAUAN SERIBU SELATAN
Kota	KABUPATEN KEPULAUAN SERIBU
Gelongan Darah	B-
Agama	Islam
Status Perkawinan	Kawin
Pekerjaan	Penduduk Pekerjaan
Kewarganegaraan	WNI
Status Kematian	Hidup

NONAKTIFKAN

Dalam membuat fitur ini saya terlebih dahulu membuat PendudukMapper yang akan berisi query mengenai pengambilan data penduduk yang dibutuhkan dari database. Berikut method-method yang saya buat dalam mapper untuk fitur ini,

Disa Ainun Yolanna

1506689654

APAP A

```
package com.example.tppapapl.dao;

import ...

@Mapper
public interface PendudukMapper {
    @Select("select * from penduduk where nik = #{nik}")
    @Results(value = {
        @Result(property = "nik", column = "nik"),
        @Result(property = "nama", column = "nama"),
        @Result(property = "tempat_lahir", column = "tempat_lahir"),
        @Result(property = "tanggal_lahir", column = "tanggal_lahir"),
        @Result(property = "golongan_darah", column = "golongan_darah"),
        @Result(property = "agama", column = "agama"),
        @Result(property = "status_perkawinan", column = "status_perkawinan"),
        @Result(property = "pekerjaan", column = "pekerjaan"),
        @Result(property = "is_wni", column = "is_wni"),
        @Result(property = "is_wafat", column = "is_wafat"),
        @Result(property = "keluarga", column = "id_keluarga",
            javaType = KeluargaModel.class,
            one = @One(select = "selectKeluarga"))
    })
    PendudukModel selectPenduduk (@Param("nik") String nik);

    @Select("select * from keluarga where id = #{id_keluarga}")
    @Results(value= {
        @Result(property = "alamat", column ="alamat"),
        @Result(property = "rt", column ="rt"),
        @Result(property = "rw", column ="rw"),
        @Result(property = "kelurahan", column = "id_kelurahan",
            javaType = KelurahanModel.class,
            one = @One(select = "selectKelurahan"))
    })
    KeluargaModel selectKeluarga (@Param("id_keluarga") String id_keluarga);
}
```

Method selectPenduduk digunakan untuk mengambil informasi mengenai penduduk yaitu, nik, nama, tempat lahir, tanggal lahir, golongan darah, agama, status perkawinan, pekerjaan, kewarganegaraan, status kematian, dan alamat. Untuk dapat mengambil alamat, saya menggunakan method selectKeluarga yang mana menggunakan id_keluarga yang saya simpan dalam atribut keluarga PendudukModel untuk mendapatkan informasi alamat, rt, rw.

```
@Select("select * from kelurahan where id = #{id_kelurahan}")
@Results(value= {
    @Result(property = "nama_kelurahan", column ="nama_kelurahan"),
    @Result(property = "kecamatan", column = "id_kecamatan",
        javaType = KecamatanModel.class,
        one = @One(select = "selectKecamatan"))
})

KelurahanModel selectKelurahan (@Param("id_kelurahan") String id_kelurahan);

@Select("select * from kecamatan where id = #{id_kecamatan}")
@Results(value= {
    @Result(property = "nama_kecamatan", column ="nama_kecamatan"),
    @Result(property = "kode_kecamatan", column ="kode_kecamatan"),
    @Result(property = "kota", column = "id_kota",
        javaType = KotaModel.class,
        one = @One(select = "selectKota"))
})
KecamatanModel selectKecamatan (@Param("id_kecamatan") String id_kecamatan);
```

Kemudian untuk mencari nama kelurahan tempat penduduk tersebut, saya menggunakan selectKeluahan. Selain mendapat nama kelurahan dari method tersebut saya juga mendapatkan id_kecamatan yang akan digunakan untuk method selectKecamatan.

Disa Ainun Yolanna
1506689654
APAP A

```
@Select("select * from kota where id = #{id_kota}")
@Results(value= {
    @Result(property = "nama_kecamatan", column = "nama_kecamatan")
})

KotaModel selectKota (@Param("id_kota") String id_kota);
```

Dari method selectKecamatan, saya bisa mendapatkan informasi nama kecamatan dan id_kota, id_kota akan digunakan oleh method selectKota untuk mendapatkan nama kota tempat tinggal penduduk tersebut.

Setelah membuat mapper saya membuat PendudukService yang akan berisi method sebagai berikut

```
package com.example.tppapap1.service;

import com.example.tppapap1.model.PendudukModel;
import java.util.List;

public interface PendudukService {
    PendudukModel selectPenduduk (String nik);
```

Kemudian saya juga membuat method di PendudukServiceDatabase yang mengimplement PendudukService dan akan memanggil method selectPenduduk yang berada di PendudukMapper

```
package com.example.tppapap1.service;

import ...

@Slf4j
@Service
public class PendudukServiceDatabase implements PendudukService{
    @Autowired
    private PendudukMapper pendudukMapper;

    @Override
    public PendudukModel selectPenduduk(String nik) {
        log.info ("select penduduk with nik {}", nik);
        return pendudukMapper.selectPenduduk(nik);
    }
}
```

Dilanjutkan dengan membuat 2 method pada Controller yaitu method index yang akan menghandle request "/" yaitu berisi halaman utama. Kemudian terdapat method searchPenduduk yang meresponse request yang dikirim dari halaman index sebelumnya dan mengembalikan path "/penduduk?{NIK}". Method searchPenduduk akan menerima parameter yaitu NIK berbentuk String, saya menginisiasi penduduk yang telah dipilih oleh Mapper ke penduduk. Apabila selectPenduduk menemukan penduduk sesuai NIK, maka penduduk akan di tampilkan ke view-penduduk.html, jika method tersebut tidak menemukan penduduk sesuai nik maka akan mengembalikan halaman penduduk-not-found.html

Disa Ainun Yolanna

1506689654

APAP A

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

@RequestMapping(method = RequestMethod.GET, value="/penduduk")
public String searchPenduduk (
    @RequestParam(value = "nik", required = true) String nik,
    Model model)
{
    PendudukModel penduduk = pendudukDAO.selectPenduduk(nik);
    if( penduduk != null) {
        model.addAttribute(attributeName: "penduduk", penduduk);
        return "view-penduduk";
    }else{
        model.addAttribute(attributeName: "nik", nik);
        return "penduduk-not-found";
    }
}
```

2. Fitur Tampilkan Data Keluarga Berdasarkan NKK

Lihat Data Keluarga Berdasarkan NKK

Masukan Nomor KK

Masukan nomor NKK

LIHAT >

Fitur ini akan menampilkan data keluarga jika user mengisi NKK keluarga tersebut dalam field yang tersedia. Ketika tombol Lihat di klik maka request akan dikirim dan mengembalikan halaman yang berisi data keluarga lengkap dengan penduduk yang menjadi anggota keluarga tersebut.

Contoh scenario fitur:

Pengguna memasukan NKK: 3171010806000002

Lihat Data Keluarga Berdasarkan NKK

Masukan Nomor KK

3171010806000002

LIHAT >

Maka akan mengembalikan halaman sebagai berikut

Disa Ainun Yolanna
1506689654
APAP A

No.	Nama Lengkap	NIK	Jenis Kelamin	Tempat Lahir	Tanggal Lahir	Agama	Pekerjaan	Status Perkawinan	Status dalam Keluarga	Kewarganegaraan
1	Salimah Nasiyidah S.Pd	3171015711880001	Perempuan	Jakarta	Thu Nov 17 00:00:00 ICT 1988	Kristen	TUKANG KAYU	Kawin	Istri	WNI
2	Pardi Simanjuntak	3171010805940001	Laki-laki	Jakarta	Sun May 08 00:00:00 ICT 1994	Kristen	PEMBANTU RUMAH TANGGA	Kawin	Kepala Keluarga	WNI

Dalam membuat fitur ini saya terlebih dahulu membuat KeluargaMapper yang akan berisi query mengenai pengambilan data keluarga yang dibutuhkan dari database. Berikut method-method yang saya buat dalam mapper untuk fitur ini,

```
@Mapper
public interface KeluargaMapper {
    @Select("Select * from keluarga where nomor_kk = #{nkk}")
    @Results(value = {
        @Result(property = "nomor_kk", column = "nomor_kk"),
        @Result(property = "alamat", column = "alamat"),
        @Result(property = "rt", column = "rt"),
        @Result(property = "rw", column = "rw"),
        @Result(property = "kelurahan", column = "id_kelurahan",
                javaType = KelurahanModel.class,
                one = @One(select = "selectKelurahan")),
        @Result(property = "penduduk", column = "nomor_kk",
                javaType = List.class, many = @Many(select = "selectPenduduk")))
    })
    KeluargaModel selectKeluarga (@Param("nkk") String nkk);

    @Select("select * from kelurahan where id = #{id_kelurahan}")
    @Results(value = {
        @Result(property = "nama_kelurahan", column = "nama_kelurahan"),
        @Result(property = "kecamatan", column = "id_kecamatan",
                javaType = KecamatanModel.class,
                one = @One(select = "selectKecamatan")))
    })

    KelurahanModel selectKelurahan (@Param("id_kelurahan") String id_kelurahan);

    @Select("select * from kecamatan where id = #{id_kecamatan}")
    @Results(value = {
        @Result(property = "nama_kecamatan", column = "nama_kecamatan"),
        @Result(property = "kota", column = "id_kota",
                javaType = KotaModel.class,
                one = @One(select = "selectKota")))
    })
    KecamatanModel selectKecamatan (@Param("id_kecamatan") String id_kecamatan);
}
```

Method selectKeluarga digunakan untuk mengambil informasi mengenai keluarga yaitu, nomor_kk, alamat, rt, dan rw. Untuk dapat mengambil nama kelurahan saya menggunakan selectKelurahan. Selain mendapat nama kelurahan dari method tersebut saya juga mendapatkan id_kecamatan yang akan digunakan untuk method selectKecamatan

```
@Select("select * from kota where id = #{id_kota}")
@Results(value = {
    @Result(property = "nama_kecamatan", column = "nama_kecamatan")
})

KotaModel selectKota (@Param("id_kota") String id_kota);
```

Disa Ainun Yolanna

1506689654

APAP A

Dari method selectKecamatan, saya bisa mendapatkan informasi nama kecamatan dan id_kota, id_kota akan digunakan oleh method selectKota untuk mendapatkan nama kota tempat tinggal penduduk tersebut.

Untuk mendapatkan informasi penduduk dari keluarga tersebut saya membuat method selectPenduduks yang akan mengembalikan List of Penduduk, jadi berisi penduduk yang merupakan bagian dari anggota keluarga tersebut. Pencarian dilakukan dengan memanfaatkan nomor_kk dan keterkaitan atribut id_keluarga dari Penduduk dan id dari Keluarga.

```
@Select("Select * from penduduk p join keluarga k on p.id_keluarga = k.id where k.nomor_kk = #{nkk} ")
@Results(value = {
    @Result(property = "nik", column = "nik"),
    @Result(property = "nama", column = "nama"),
    @Result(property = "tempat_lahir", column = "tempat_lahir"),
    @Result(property = "tanggal_lahir", column = "tanggal_lahir"),
    @Result(property = "jenis_kelamin", column = "jenis_kelamin"),
    @Result(property = "golongan_darah", column = "golongan_darah"),
    @Result(property = "agama", column = "agama"),
    @Result(property = "status_perkawinan", column = "status_perkawinan"),
    @Result(property = "pekerjaan", column = "pekerjaan"),
    @Result(property = "status_dalam_keluarga", column = "status_dalam_keluarga"),
    @Result(property = "is_wni", column = "is_wni")
})
List<PendudukModel> selectPenduduks (@Param("nkk") String nkk);
```

Setelah membuat mapper saya membuat KeluargaService yang akan berisi method sebagai berikut dengan parametetr NKK

```
import ...

public interface KeluargaService {
    KeluargaModel selectKeluarga (String nkk);
```

Kemudian saya juga membuat method di KeluargaServiceDatabase yang mengimplement KeluargaService dan akan memanggil method selectKeluarga yang berada di KeluargaMapper dengan menginisiasikan keluargaMapper sebelumnya.

```
@Slf4j
@Service
public class KeluargaServiceDatabase implements KeluargaService {
    @Autowired
    private KeluargaMapper keluargaMapper;

    public KeluargaModel selectKeluarga(String nkk) {
        log.info("select keluarga with nkk {}", nkk);
        return keluargaMapper.selectKeluarga(nkk);
    }
}
```

Dilanjutkan dengan membuat method pada KeluargaController, terdapat method searchKeluarga yang meresponse request yang dikirim dari halaman index sebelumnya dan mengembalikan path “/keluarga?{NKK}”. Method searchKeluarga akan menerima parameter yaitu NKK berbentuk String, saya menginisiasikan keluarga yang telah dipilih oleh KeluargaMapper ke object keluarga. Apabila selectKeluarga menemukan keluarga sesuai NKK, maka keluarga akan di tampilkan ke view-keluarga.html, jika method

Disa Ainun Yolanna

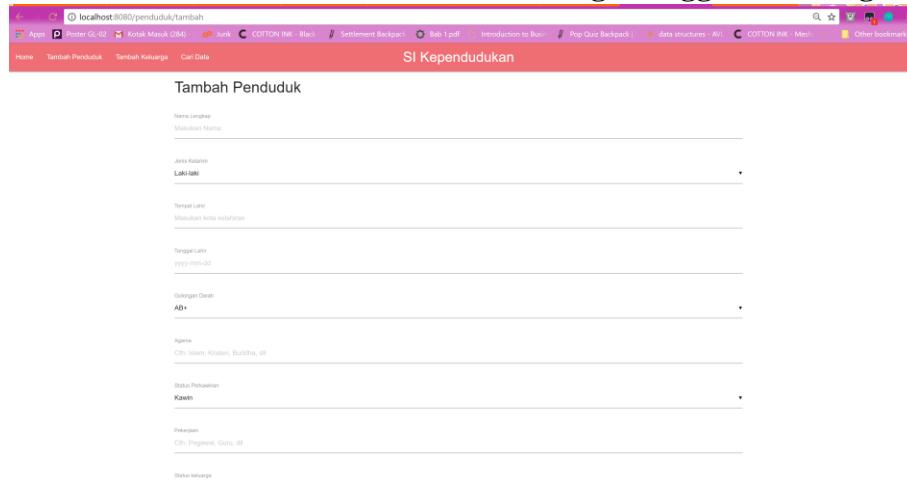
1506689654

APAP A

tersebut tidak menemukan penduduk sesuai nik maka akan mengembalikan halaman keluarga-not-found.html

```
@RequestMapping(method = RequestMethod.GET, value="/keluarga")
public String searchKeluarga (
    @RequestParam(value = "nkk", required = true) String nkk,
    Model model)
{
    KeluargaModel keluarga = keluargaDAO.selectKeluarga(nkk);
    if( keluarga != null) {
        model.addAttribute(attributeName: "keluarga", keluarga);
        return "view-keluarga";
    }else{
        model.addAttribute(attributeName: "nkk", nkk);
        return "keluarga-not-found";
    }
}
```

3. Fitur Menambahkan Penduduk Baru sebagai Anggota Keluarga



The screenshot shows a web application interface for adding a new resident. The title bar says 'SI Kependudukan'. The main heading is 'Tambah Penduduk'. The form contains the following fields:

- Name: Name Length: Masukkan Nama
- Gender: Jenis Kelamin: Laki-laki
- Birthplace: Tempat Lahir: Masukkan Alamat Kelahiran
- Birthdate: Tanggal Lahir: yyyy-mm-dd
- Religion: Golongan Darah: AB+
- Agama: Ctr: Islam, Kristen, Buddha, dkk
- Status Perkawinan: Kawin
- Nationality: Negara: Ctr: Pergres, Gres, dkk

Fitur ini akan menambahkan data penduduk dan menambahkan penduduk sebagai anggota keluarga dari keluarga yang telah terdaftar dalam database. Pengguna perlu mengisi form untuk menambahkan informasi mengenai penduduk tersebut. Kemudian mengklik tombol submit yang ada di bawah form. Setiap penduduk yang ditambahkan akan digenerate NIKnya.

Contoh scenario fitur 3:

Pengguna memasukan informasi mengenai penduduk dengan mengisi form (asumsi pengguna mengisi form dengan benar dan valid)

Disa Ainun Yolanna
1506689654
APAP A

SI Kependudukan

Tambah Penduduk

Nama Lengkap
Johnson V

Jenis Kelamin
Laki-laki

Tempat Lahir
Palembang

Tanggal Lahir
1990-07-17

Golongan Darah
B+

Agama
Islam

Status Perkawinan
Kawin

Pekerjaan
Pegawai

Status Keluarga
Anak

Kewarganegaraan
WNI

Status Kematian
Hidup

ID Keluarga
63406

SUBMIT >

Setelah pengguna mengklik tombol submit maka akan muncul tampilan



Sukses!

Penduduk dengan NIK - 3171021707900001 berhasil ditambahkan

Hal yang pertama saya lakukan yaitu dengan membuat method addPenduduk di PendudukMapper. Yang berisi query insert mengenai data-data yang dicantumkan dalam form

```
@Insert("insert into penduduk (nik, nama, tempat_lahir, tanggal_lahir, jenis_kelamin, golongan_darah, agama, status_perkawinan, pekerjaan, is_wni, is_wafat, id_keluarga, status_dalam_keluarga) " +  
"values (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"  
void addPenduduk(@Param("penduduk") PendudukModel penduduk, @Param("id_keluarga") String id_keluarga);
```

Setelah itu, saya membuat method addPenduduk dalam PendudukService yang akan menerima model penduduk dan id_keluarga.

```
void addPenduduk (PendudukModel penduduk, String id_keluarga);
```

Disa Ainun Yolanna

1506689654

APAP A

Method dibawah merupakan method addPenduduk yang berada di PendudukServiceDatabase yang akan menerima parameter penduduk dan id_keluarga. Method inilah yang akan mengatur NIK dari method generateNIK yang tersedia di PendudukServiceDatabase. Setelah mengatur NIK method addPenduduk juga akan memanggil pendudukMapper untuk menambahkan penduduk yang telah memiliki nik dan informasi lainnya lengkap dengan id_keluarga ke database.

```
@Override  
public void addPenduduk(PendudukModel penduduk, String id_keluarga)  
{  
    penduduk.setNik(generateNIK(penduduk, id_keluarga));  
    pendudukMapper.addPenduduk(penduduk, id_keluarga);  
}
```

Berikut dibawah ini merupakan method generateNIK penduduk yang akan menghasilkan NIK dari informasi penduduk yang diinput oleh user. 16 digit NIK sendiri terdiri dari domisili, tanggal lahir, dan nomor urut. Dalam method generateNIK, saya membutuhkan parameter penduduk dan id_keluarga.

```
public String generateNIK(PendudukModel penduduk, String id_keluarga) {  
    String digit= digitDT(penduduk,id_keluarga) + '%';  
    List<String> niks = pendudukMapper.selectNIKs(digit);  
    digit = digit.substring(0,digit.length()-1);  
    if(niks.size() == 0){  
        digit += "0001";  
    }else{  
        Long nikMAX= (long) 0;  
        for (int i=0;i<niks.size();i++){  
            Long tmp = Long.parseLong(niks.get(i));  
            if(nikMAX < tmp){  
                nikMAX = tmp;  
            }  
        }  
        digit = Long.toString(i:nikMAX + 1);  
    }  
    log.info("nik {}",digit);  
    return digit;  
}
```

Pertama saya menggenrate 6 digit awal dengan memanggil method digitDT yang akan mengambil domisili tempat tinggal penduduk dari id_keluarga dan 6 digit tanggal lahir dari data yang dimasukan di form. Untuk mendapatkan 6 digit domisili, method ini dapatkan dari kodeKecamatan dari tempat tinggal keluarga penduduk. Untuk tanggal lahir hanya tinggal mengubah format string dari form ke date untuk bisa masuk ke database. Sebelum itu apabila penduduk tersebut berjenis kelamin perempuan maka akan terjadi penambahan 40 pada angka-angka tanggal lahirnya. Setelah mendapat 12 digit pertama, Method selectIdKeluarga akan digunakan di method digitDT untuk menemukan

Disa Ainun Yolanna

1506689654

APAP A

domisili.

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

public String digitDT(PendudukModel penduduk, String id_keluarga){
    log.info("Select keluarga with id_keluarga {}", id_keluarga);
    KeluargaModel keluarga = pendudukMapper.selectKeluarga(id_keluarga);
    String digital = keluarga.getKelurahan().getKecamatan().getKode_kecamatan().substring(0, 6);
    DateFormat df = new SimpleDateFormat(pattern: "ddMMyy");
    String tgllahir = df.format(penduduk.getTanggal_lahir());
    if(penduduk.getJenis_kelamin() == 1{
        int tgl = Integer.parseInt(tgllahir.substring(0,2));
        int res = tgl + 40;
        String newdigit = Integer.toString(res);
        tgllahir = newdigit + tgllahir.substring(2);
    }
    String digit = digit1 + tgllahir;
    return digit;
}
```

4 digit terakhir akan dilanjutkan oleh method generateNIK. NIK memiliki nilai yang unik. Untuk dapat mengetahui apakah terdapat penduduk yang memiliki domisili dan tanggal lahir yang sama, maka method ini akan memanggil method selectNIKs dari PendudukMapper untuk menemukan NIK yang mirip.

```
@Select("select nik from penduduk where nik like #{digit}")
List<String> selectNIKs(@Param("digit") String digit);
```

Jika terdapat digit awal NIK yang sama maka NIK tersebut tidak diakhiri oleh ‘0001’ tapi bertambah dari angka terakhir yang telah ada sebelumnya di database. Cara mengetahuinya adalah dengan membandingkannya.

Setelah menemukan mengeset NIK, akan dilanjutkan dengan membuat method dicontroller sebagai pengeksekusi

Method addKeluarga akan membuat Penduduk baru dan menjalankan ‘GET’ dengan menampilkan form-penduduk dan mengambil value-value dalam field. Method addSubmit akan menerima request dari ‘GET’ dengan menerima requestparam id_keluarga, tgl_lahir dan penduduk. Dalam method ini saya mengubah format tanggal lahir dari String ke Date untuk bisa diset ke penduduk baru. Dilanjutkan dengan menampilkan pesan penduduk sukses dibuat dengan menghadirkan NIK yang telah ada

Disa Ainun Yolanna
1506689654
APAP A

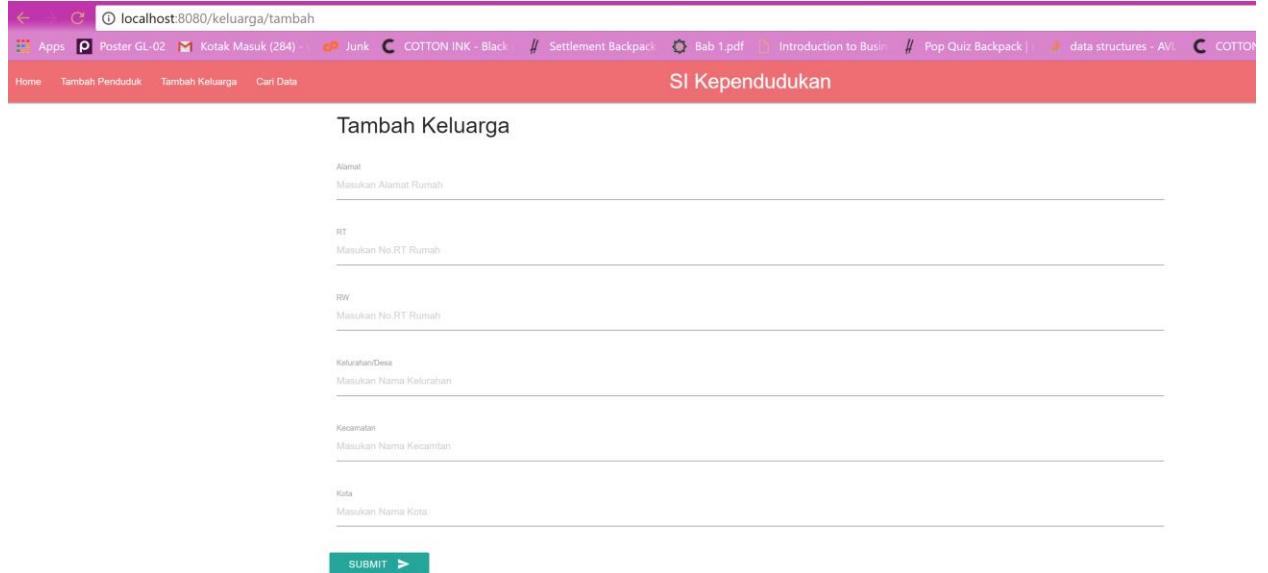
```
@RequestMapping(method = RequestMethod.GET, value = "/penduduk/tambah")
public String addPenduduk(Model model) {
    PendudukModel penduduk = new PendudukModel();
    model.addAttribute(attributeName: "penduduk", penduduk);
    return "form-penduduk";
}

@RequestMapping(method = RequestMethod.POST, value="/penduduk/tambah")
public String addSubmit (
    @RequestParam (value = "id_keluarga", required = false) String id_keluarga,
    @RequestParam(value = "tgl_lahir", required = false) String tgl_lahir,
    Model model, PendudukModel penduduk) throws Exception

{

    Date date = new SimpleDateFormat(pattern: "yyyy-MM-dd").parse(tgl_lahir);
    penduduk.setTanggal_lahir(date);
    pendudukDAO.addPenduduk(penduduk, id_keluarga);
    model.addAttribute(attributeName: "penduduk", penduduk);
    return "penduduk-success-add";
}
```

4. Fitur Menambah Keluarga Baru



The screenshot shows a web application interface for adding a new family. The browser address bar displays "localhost:8080/keluarga/tambah". The page title is "SI Kependudukan". Below the title, there is a navigation menu with links: Home, Tambah Penduduk, Tambah Keluarga, and Cari Data. The main content area is titled "Tambah Keluarga". It contains five input fields with labels and placeholders:

- Alamat: Masukan Alamat Rumah
- RT: Masukan No.RT Rumah
- RW: Masukan No.RW Rumah
- Kecamatan: Masukan Nama Kecamatan
- Kota: Masukan Nama Kota

At the bottom of the form is a green "SUBMIT" button with a right-pointing arrow.

Fitur ini akan menambahkan data keluarga dan menambahkan keluarga ke dalam database. Pengguna perlu mengisi form untuk menambahkan informasi mengenai keluarga tersebut. Kemudian mengklik tombol submit yang ada di bawah form. Setiap keluarga yang ditambahkan akan digenerate NKKnya.

Disa Ainun Yolanna

1506689654

APAP A

Contoh Scenario

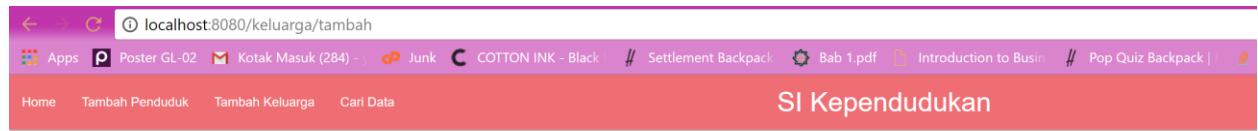
Pengguna menambahkan keluarga dengan mengisi form (asumsi pengguna mengisi form dengan benar dan valid)

The screenshot shows a web browser window with the URL `localhost:8080/keluarga/tambah`. The page title is "SI Kependudukan". The main content is a form titled "Tambah Keluarga". The form has several input fields:

- Alamat: Jalan Belanda
- RT: 11
- RW: 1
- Kelurahan/Desa: KEBAGUSAN
- Kecamatan: Pasar Minggu
- Kota: Kota Jakarta Selatan

A green "SUBMIT" button is located at the bottom of the form.

Setelah mengsubmit form, maka akan menampilkan halaman



Hal yang pertama saya lakukan yaitu dengan membuat method addKeluarga di KeluargaMapper. Yang berisi query insert mengenai data-data yang dicantumkan dalam form

```
@Insert("insert into keluarga (nomor_kk,alamat, rt, rw, id_kelurahan) " +  
        "values (#{keluarga.nomor_kk},#{keluarga.alamat},#{keluarga.rt},#{keluarga.rw},#{id_kelurahan})")  
void addKeluarga (@Param("keluarga") KeluargaModel keluarga, @Param("id_kelurahan") String id_kelurahan);
```

Setelah itu, saya membuat method addKeluarga dalam KeluargaService yang akan menerima model keluarga dan id_kelurahan.

```
void addKeluarga (KeluargaModel keluarga, String id_kelurahan);
```

Pada kasus ini, saya membutuhkan id_kelurahan untuk dapat mencari kecamatan dan kota yang valid. Hal ini dikarenakan pada form-keluarga-add saya menggunakan input pada kolom kelurahan, kota, kecamatan, sehingga program ini hanya dapat menggunakan kelurahan agar valid untuk menemukan nilai kolom lainnya yang tidak terambil nilainya. Method dibawah merupakan method addKeluarga yang berada di KeluargaServiceDatabase yang akan menerima parameter keluarga dan id_kelurahan. Method inilah yang akan mengatur NKK dari method generateNKK yang tersedia di KeluargaServiceDatabase. Setelah mengatur NKK method addKeluarga juga akan

Disa Ainun Yolanna

1506689654

APAP A

memanggil keluargaMapper untuk menambahkan keluarga yang telah memiliki nkk dan informasi lainnya lengkap dengan domisili ke database.

```
public void addKeluarga(KeluargaModel keluarga, String id_kelurahan){  
    keluarga.setNomor_kk(generateNKK(keluarga, id_kelurahan));  
    keluargaMapper.addKeluarga(keluarga, id_kelurahan);  
}  
  
@Override  
public String selectIdKelurahan(String nama_kelurahan) {  
    return keluargaMapper.selectIdKelurahan(nama_kelurahan);  
}
```

Berikut dibawah ini merupakan method generateNKK keluarga yang akan menghasilkan NKK dari informasi keluarga yang diinput oleh user. 16 digit NKK sendiri terdiri dari domisili, tanggal input data, dan nomor urut. Dalam method generateNKK, saya membutuhkan parameter keluarga dan id _kelurahan.

```
public String generateNKK(KeluargaModel keluarga, String id_kelurahan){  
    String digit = digitDT(keluarga,id_kelurahan) + "1";  
    List<String> nkks = keluargaMapper.selectNKKs(digit);  
    digit = digit.substring(0,digit.length()-1);  
    if (nkks.size() == 0){  
        digit += "0001";  
    }else{  
        Long nkkMAX= (long) 0;  
        for (int i=0;i<nkks.size();i++){  
            Long tmp = Long.parseLong(nkks.get(i));  
            if(nkkMAX < tmp){  
                nkkMAX = tmp;  
            }  
        }  
        digit = Long.toString(i:nkkMAX + 1);  
    }  
    log.info("nkk {}",digit);  
    return digit;  
}
```

Pertama saya menggenrate 6 digit awal dengan memanggil method digitDT yang akan mengambil domisili tempat tinggal keluarga berdasarkan nama kelurahan yang diinput menggunakan method selectKelurahan dalam keluarga mapper, dengan mendapatkan id_kelurahan kita dapat mengetahui informasi mengenai kota dan kecamatan. Kemudian 6 digit tanggal dari data dimasukan di form. Untuk mendapatkan 6 digit domisili, method ini dapatkan dari kodeKecamatan dari id_kelurahan yang ditemukan sebelumnya. Untuk tanggal data diinput hanya tinggal mengubah format string dari form ke date untuk bisa masuk ke database.

Disa Ainun Yolanna
1506689654
APAP A

```
public String digitDT(KeluargaModel keluarga, String id_kelurahan){  
    log.info("Select kelurahan with id_kelurahan {}", id_kelurahan);  
    KelurahanModel kelurahan = keluargaMapper.selectKelurahan(id_kelurahan);  
    log.info("kelurahan", id_kelurahan);  
  
    String digit1 = kelurahan.getKecamatan().getKode_kecamatan().substring(0, 6);  
    log.info("digit1", digit1);  
    DateFormat dateFormat = new SimpleDateFormat(pattern: "ddMMyy");  
    Date date = new Date();  
    String date2 = dateFormat.format(date);  
    String digit = digit1 + date2 ;  
    return digit;  
}
```

4 digit terakhir akan dilanjutkan oleh method generateNKK. NKK memiliki nilai yang unik. Untuk dapat mengetahui apakah terdapat keluarga yang memiliki domisili dan tanggal input data yang sama, maka method ini akan memanggil method selectNKKs dari KeluargaMapper untuk menemukan NKK yang mirip.

```
@Select("select nomor_kk from keluarga where nomor_kk like #{digit}")  
List<String> selectNKKs(@Param("digit") String digit);
```

Jika terdapat digit awal NKK yang sama maka NKK tersebut tidak diakhiri oleh ‘0001’ tapi bertambah dari angka terakhir yang telah ada sebelumnya di database. Cara mengetahuinya adalah dengan membandingkannya di method generateNKK.

Setelah menemukan mengeset NKK, akan dilanjutkan dengan membuat method diKeluargaController sebagai pengeksekusi

Method addKeluarga akan membuat Keluarga baru dan menjalankan ‘GET’ dengan menampilkan form-keluarga dan mengambil value-value dalam field. Method addSubmit akan menerima request dari ‘GET’ dengan menerima requestparam nama_kelurahan dan keluarga,model. Dalam method ini saya memanggil method addKeluarga berisikan parameter yang dibutuhkan.

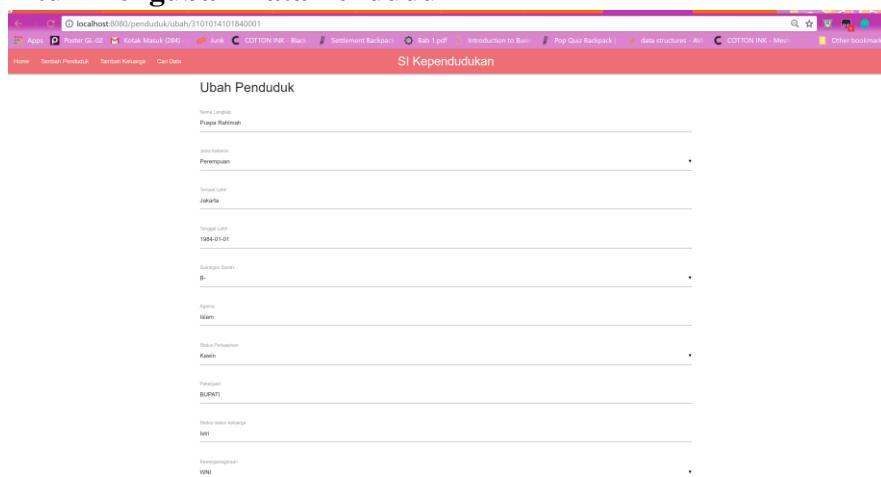
```
@RequestMapping(method = RequestMethod.GET, value = "/keluarga/tambah")  
public String addKeluarga(Model model){  
    KeluargaModel keluarga = new KeluargaModel();  
    model.addAttribute(attributeName: "keluarga", keluarga);  
    return "form-keluarga";  
}  
  
@RequestMapping(method = RequestMethod.POST, value="/keluarga/tambah")  
public String addSubmit (  
    @RequestParam (value = "nama_kelurahan", required = false) String nama_kelurahan,  
    Model model, KeluargaModel keluarga) throws Exception  
  
{  
    keluargaDAO.addKeluarga(keluarga, keluargaDAO.selectIdKelurahan(nama_kelurahan));  
    model.addAttribute(attributeName: "keluarga", keluarga);  
    return "keluarga-success-add";  
}
```

Disa Ainun Yolanna

1506689654

APAP A

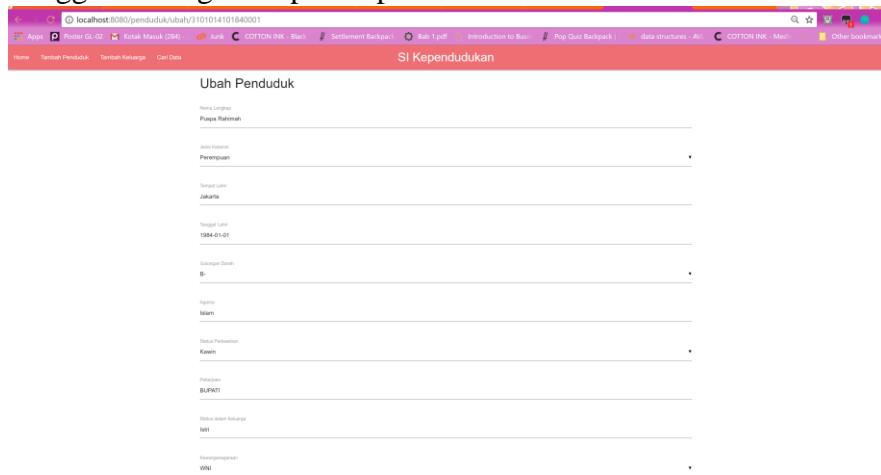
5. Fitur Mengubah Data Penduduk



Fitur ini akan mengubah data penduduk dengan mengembalikan form yang telah berisi informasi lengkap penduduk yang dapat diubah. Dalam fitur ini, pengguna dapat mempengaruhi NIK jika mengubah id_keluarga dan tanggal lahir penduduk

Contoh Skenario

Pengguna mengakses path “/penduduk/ubah/3101014101840001”



Mengubah tanggal lahir penduduk tersebut. Kemudian mengklik submit
Maka akan menampilkan halaman berikut



Sukses!

Penduduk dengan NIK - 3101014101840001 berhasil diubah

Yang berisikan nik lama penduduk tersebut, namun didatabase nik akan berubah karena perubahan tanggal lahir.

Fitur ini dimulai dengan pembuatan method updatePenduduk di PendudukMapper, untuk dapat mengupdate data penduduk tersebut. Method ini menerima parameter nik, penduduk dan id_keluarga.

```
@Update("update penduduk set nik = #{penduduk.nik}, nama = #{penduduk.nama}, tempat_lahir = #{penduduk.tempat_lahir}, tanggal_lahir = #{penduduk.tanggal_lahir}, jenis_kelamin = #{penduduk.jenis_ke_is_wln = #{penduduk.is_wln}, id_keluarga = #{id_keluarga}, agama= #{penduduk.agama}, pekerjaan = #{penduduk.pekerjaan}, status_perkawinan = #{penduduk.status_perkawinan}, status_dalam_ke_is_wafat = #{penduduk.is_wafat} where nik = #{nik})")  
void updatePenduduk(@Param("nik") String nik, @Param("penduduk") PendudukModel penduduk, @Param("id_keluarga") String id_keluarga);
```

Kemudian dilanjutkan dengan membuat method updatePenduduk di PendudukService

```
void updatePenduduk (String nik, PendudukModel penduduk, String id_keluarga);
```

Dan di override di PendudukServiceDatabase. Method dibawah ini akan menerima value baru dari form dan akan mengecek apabila terdapat perubahan pada domisili dan tanggal lahir dari penduduk. Karena 2 hal itulah yang mempengaruhi perubahan NIK. Dengan membandingkan, apabila tidak sama, maka NIK baru harus di generate ulang dengan menggunakan method yang sama sebelumnya dalam menambah penduduk. Kemudian method akan memanggil pendudukMapper untuk memasukan informasi bar uke database.

```
@Override  
public void updatePenduduk(String nik, PendudukModel penduduk, String id_keluarga) {  
  
    String newDigitDT= digitDT(penduduk,id_keluarga);  
    String oldDigitDT = nik.substring(0,12);  
    if(!newDigitDT.equals(oldDigitDT)){  
        penduduk.setNik(generateNIK(penduduk,id_keluarga));  
    }  
    pendudukMapper.updatePenduduk(nik,penduduk,id_keluarga);  
}
```

```
@Override
```

Pada method controller dibawah ini yaitu updatePenduduk, akan mengambil penduduk dengan nik yang terdapat dalam variable path, kemudian apabila penduduk ditemukan

Disa Ainun Yolanna

1506689654

APAP A

berarti penduduk tersebut dapat diubah informasinya. Method ini akan mengembalikan form yang berisi data penduduk apabila penduduk berhasil ditemukan

```
@RequestMapping(method = RequestMethod.GET, value = "/penduduk/ubah/{nik}")
public String updatePenduduk(@PathVariable (value = "nik") String nik, Model model) {
    PendudukModel penduduk = pendudukDAO.selectPenduduk(nik);
    if (penduduk != null) {
        DateFormat df = new SimpleDateFormat (pattern: "yyyy-MM-dd");
        String tgl_lahir = df.format(penduduk.getTanggal_lahir());
        model.addAttribute(attributeName: "tgl_lahir", tgl_lahir);

        String goldar = penduduk.getGolongan_darah();
        if(goldar.charAt(goldar.length()-1) == '-'){
            penduduk.setGolongan_darah(goldar.substring(0,goldar.length()-1)+"-");
        }

        String id_keluarga = pendudukDAO.selectIdKeluarga(nik);
        model.addAttribute(attributeName: "id_keluarga", id_keluarga);

        model.addAttribute(attributeName: "penduduk", penduduk);
        return "form-penduduk-update";
    }
    model.addAttribute(attributeName: "nik", nik);
    return "not-found";
}
```

Berikut form-penduduk-update yang menggunakan th:value, th:text untuk menampilkan data penduduk dari database.

```
<br/>
<div class="row">
    <form class="col s12" th:object="${penduduk}" th:action="/penduduk/ubah/' + ${penduduk.nik}>
        <div class="row">
            <div class="input-field col s12">
                <input th:value="${nama}" th:field="*{nama}" placeholder="Masukan Nama" id="name">
                <label for="name">Nama Lengkap</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <select th:value="${jenis_kelamin}" th:field="*{jenis_kelamin}">
                    <option th:value="3" disabled="disabled" selected="selected">Pilih Jenis Kelamin</option>
                    <option th:value="1">Perempuan</option>
                    <option th:value="0">Laki-laki</option>
                </select>
                <label>Jenis Kelamin</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <input th:value="${tempat_lahir}" th:field="*{tempat_lahir}" placeholder="Masukkan Tempat Lahir" id="tempat_lahir">
                <label for="tempat_lahir">Tempat Lahir</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <input th:value="${tgl_lahir}" name="tgl_lahir" placeholder="yyyy-mm-dd" id="tgl_lahir">
                <label for="tgl_lahir">Tanggal Lahir</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <select th:field="*{golongan_darah}">
                    <option value="0" disabled="disabled" selected="selected">Pilih Status Golongan Darah</option>
                    <option value="AB+>AB+</option>
                    <option value="AB->AB-</option>
                </select>
            </div>
        </div>
    </form>
</div>
```

Disa Ainun Yolanna
1506689654
APAP A

6. Fitur Mengubah Data Keluarga

Ubah Keluarga

Alamat
Jln. Tangkuban Perahu No. 767

RT
155

RW
085

Kelurahan/Desa
CIPEDAK

Kecamatan
JAGAKARSA

Kota
KOTA JAKARTA SELATAN

SUBMIT >

Fitur ini akan mengubah data keluarga dengan mengembalikan form yang telah berisi informasi lengkap keluarga yang dapat diubah. Dalam fitur ini, pengguna dapat mempengaruhi NKK jika mengubah kelurahan dan mengisi data dengan tanggal yang berbeda dari sebelumnya.

Contoh Skenario

Pengguna mengakses path “/penduduk/ubah/ 3171010804990001”

Ubah Keluarga

Alamat
Jln. Tangkuban Perahu No. 767

RT
155

RW
085

Kelurahan/Desa
CIPEDAK

Kecamatan
JAGAKARSA

Kota
KOTA JAKARTA SELATAN

SUBMIT >

Pengguna mengubah kelurahan menjadi kebagusan, Kemudian mengklik submit Maka akan menampilkan halaman berikut (Asumsi pengguna menginput Kecamatan dan Kota sesuai dengan Kelurahan)

Disa Ainun Yolanna
1506689654
APAP A



Sukses!

Keluarga dengan NKK - 3171010804990001 berhasil diubah

Yang berisikan nkk lama keluarga tersebut, namun didatabase nkk akan berubah karena perubahan kelurahan.

Fitur ini dimulai dengan pembuatan method updateKeluarga di KeluargaMapper, untuk dapat mengupdate data keluarga tersebut. Method ini menerima parameter nkk, keluarga dan id_kelurahan.

```
@Update("update keluarga set nomor_kk = #{keluarga.nomor_kk}, alamat = #{keluarga.alamat}, rt = #{keluarga.rt}, " +  
"rw = #{keluarga.rw}, id_kelurahan= #{id_kelurahan} where nomor_kk = #{nkk}" )  
void updateKeluarga (@Param("nkk") String nkk, @Param("keluarga") KeluargaModel keluarga, @Param("id_kelurahan") String id_kelurahan);
```

Kemudian dilanjutkan dengan membuat method updateKeluarga di KeluargaService

```
void updateKeluarga(String nkk, KeluargaModel keluarga, String id_kelurahan);
```

Dan di override di KeluargaServiceDatabase. Method dibawah ini akan menerima value baru dari form dan akan mengecek apabila terdapat perubahan pada domisili dan tanggal input data dari keluarga. Karena 2 hal itulah yang mempengaruhi perubahan NKK. Dengan membandingkan, apabila tidak sama, maka NKK baru harus di generate ulang dengan menggunakan method yang sama sebelumnya dalam menambah keluarga. Kemudian method akan memanggil keluargaMapper untuk memasukan informasi bar uke database.

```
public void updateKeluarga(String nkk, KeluargaModel keluarga, String id_kelurahan) {  
    String newDigitDT = digitDT(keluarga, id_kelurahan);  
    log.info("newDigitDT {}", newDigitDT);  
    String oldDigitDT = nkk.substring(0,12);  
    if(!newDigitDT.equals(oldDigitDT)){  
        keluarga.setNomor_kk(generateNKK(keluarga, id_kelurahan));  
    }  
    keluargaMapper.updateKeluarga(nkk, keluarga, id_kelurahan);  
}
```

Pada method di KeluargaController dibawah ini yaitu updateKeluarga, akan mengambil keluarga dengan nkk yang terdapat dalam variable path, kemudian apabila keluarga

Disa Ainun Yolanna

1506689654

APAP A

ditemukan berarti keluarga tersebut dapat diubah informasinya. Method ini akan mengembalikan form yang berisi data keluarga apabila keluarga berhasil ditemukan

```
@RequestMapping(method = RequestMethod.GET, value = "/keluarga/ubah/{nkk}")
public String updatePenduduk(@PathVariable (value = "nkk") String nkk, Model model){
    KeluargaModel keluarga = keluargaDAO.selectKeluarga(nkk);
    if(keluarga != null){

        String nama_kelurahan = keluarga.getKelurahan().getNama_kelurahan();
        String nama_kecamatan = keluarga.getKelurahan().getKecamatan().getNama_kecamatan();
        String nama_kota = keluarga.getKelurahan().getKecamatan().getKota().getNama_kota();
        model.addAttribute(attributeName: "nama_kelurahan",nama_kelurahan);
        model.addAttribute(attributeName: "nama_kecamatan",nama_kecamatan);
        model.addAttribute(attributeName: "nama_kota",nama_kota);
        model.addAttribute(attributeName: "keluarga",keluarga);
        return "form-keluarga-update";
    }
    model.addAttribute(attributeName: "nkk",nkk);
    return "not-found";
}

@RequestMapping(method = RequestMethod.POST, value = "/keluarga/ubah/{nkk}")
public String updateSubmit( KeluargaModel keluarga,
                           @PathVariable (value = "nkk") String nkk,@RequestParam(value = "nama_kelurahan") String nama_kelurahan
) throws Exception {

    String id_kelurahan = keluargaDAO.selectIdKelurahan(nama_kelurahan);
    ArrayList<PendudukModel> penduduks = new ArrayList<>(keluargaDAO.selectKeluarga(nkk).getPenduduks());
    keluargaDAO.updateKeluarga(nkk, keluarga,id_kelurahan);
    if (penduduks.size()>0) {
        log.info("nik {}", penduduks.get(0).getNik());
        String id_keluarga = pendudukDAO.selectIdKeluarga(penduduks.get(0).getNik());
        for (PendudukModel penduduk : penduduks
             ) {
            pendudukDAO.updatePenduduk(penduduk.getNik(), penduduk, id_keluarga);

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

Berikut form-penduduk-update yang menggunakan th:value, th:text untuk menampilkan data penduduk dari database.

Disa Ainun Yolanna

1506689654

APAP A

```
<div>
<div class="row">
    <form class="col s12" th:object="${keluarga}" th:action="/keluarga/ubah/" + ${keluarga.nomor_kk}" method="post">
        <div class="row">
            <div class="input-field col s12">
                <input th:field="*{alamat}" th:value="${alamat}" placeholder="Masukan Alamat Rumah" id="alamat" type="text">
                <label for="alamat">Alamat</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <input th:field="*{rt}" th:value="${rt}" placeholder="Masukan No.RT Rumah" id="rtrumah" type="text">
                <label for="rtrumah">RT</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <input th:field="*{rw}" th:value="${rw}" placeholder="Masukan No.RT Rumah" id="rwrumah" type="text">
                <label for="rwrumah">RW</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <input name="nama_kelurahan" th:value="${nama_kelurahan}" placeholder="Masukan Nama Kelurahan" id="kelurahan" type="text">
                <label for="kelurahan">Kelurahan/Desa</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <input th:value="${nama_kecamatan}" placeholder="Masukan Nama Kecamatan" id="kecamatan" type="text">
                <label for="kecamatan">Kecamatan</label>
            </div>
        </div>

        <div class="row">
            <div class="input-field col s12">
                <input th:value="${nama_kota}" placeholder="Masukan Nama Kota" id="kota" type="text" class="validate">
                <label for="kota">Kota</label>
            </div>
        </div>
    </form>
</div>
```

7. Fitur Mengubah Status Kematian Penduduk

Lihat Data Penduduk -3101014102810001

NIK: 3101014102810001

Nama: Juli Utami

Tempat/Tanggal Lahir: Jakarta, Sun Feb 01 00:00:00 ICT 1981

Alamat: Ds. Djenengoro No. 927

RT/RW: 161/02

Kelurahan/Desa: PULAU PARI

Kecamatan: KEPULAUAN SERIBU SELATAN

Kota: KABUPATEN KEPULAUAN SERIBU

Golongan Darah: O-

Agama: Islam

Status Perkawinan: Belum Kawin

Pekerjaan: Penduduk Pekerjaan

Kewarganegaraan: WNI

Status Kematian: Hidup

NONAKTIFKAN

Fitur ini akan menonaktifkan penduduk karena telah wafat dengan mengklik tombol non aktif yang berada disebelah status kematian. tombol ini hanya muncul jika status

Disa Ainun Yolanna

1506689654

APAP A

penduduk tersebut masih hidup. Fitur ini juga akan menonaktifkan nkk keluarga yang anggota keluarganya telah wafat semua.

Contoh scenario:

Pengguna mengakses suatu nik, dan halaman menampilkan informasi penduduk tersebut

Lihat Data Penduduk -3101014102810001

NIK	3101014102810001
Nama	Juli Utami
Tempat/Tanggal Lahir	Jakarta, Sun Feb 01 00:00:00 ICT 1981
Alamat	Ds. Dipenogoro No. 927
RT/RW	161/092
Kelurahan/Desa	PULAU PARI
Kecamatan	KEPULAUAN SERIBU SELATAN
Kota	KABUPATEN KEPULAUAN SERIBU
Golongan Darah	O-
Agama	Islam
Status Perkawinan	Belum Kawin
Pekerjaan	Penduduk Pekerjaan
Kewarganegaraan	WNI
Status Kematian	Hidup

NONAKTIFKAN

Kemudian pengguna akan menklik non aktif jika penduduk tersebut telah wafat, maka halaman akan mengembalikan pesan sukses

Disa Ainun Yolanna

1506689654

APAP A

The screenshot shows a browser window with the URL `localhost:8080/penduduk?nik=3101014102810001`. The title bar says "SI Kependudukan". The main content area displays a success message "Sukses!" and a table of demographic information:

Atribut	Nilai
NIK	3101014102810001
Nama	Juli Utami
Tempat/Tanggal Lahir	Jakarta, Sun Feb 01 00:00:00 ICT 1981
Alamat	Ds. Digenegoro No. 927
RT/RW	161/092
Kelurahan/Desa	PULAU PARI
Kecamatan	KEPULAUAN SERIBU SELATAN
Kota	KABUPATEN KEPULAUAN SERIBU
Golongan Darah	O-
Agama	Islam
Status Perkawinan	Belum Kawin
Pekerjaan	Penduduk Pekerjaan
Kewarganegaraan	WNI
Status Kematian	Wafat

Pada fitur ini saya membuat method pada PendudukMapper yang berisi akan mengset atribut is_wafat menjadi telah wafat/true.

```
@Update("update penduduk set is_wafat = 1 where nik = #{nik}")
void pendudukWafat(@Param("nik") String nik);
```

Kemudian saya membuat method di PendudukService

```
void pendudukWafat(String nik);
```

Dan dioveride di pendudukservicedatabase

```
@Override
public void pendudukWafat(String nik) {

    log.info("niknya {}", nik);
    pendudukMapper.pendudukWafat(nik);
}
```

Disa Ainun Yolanna

1506689654

APAP A

Dan melanjutkannya di pendudukcontroller. Method ini memanggil 2 dao. Pendudukdao untuk mengset status kematian penduduk berdasarkan nik, sedangkan keluarga dao, untuk mengecek apakah nkk tetap berlaku atau tidak dengan meninggalnya anggota keluarganya.

```
@RequestMapping (value = "/penduduk/mati", method = RequestMethod.POST)
public String pendudukMati(@RequestParam(value = "nik") String nik, RedirectAttributes redirectAttributes{
    pendudukDAO.pendudukWafat(nik);
    keluargaDAO.updateIsBerlaku(pendudukDAO.selectIdKeluarga(nik));
    redirectAttributes.addFlashAttribute(attributeName: "wafat", attributeValue: "true");
    return "redirect:/penduduk?nik=" + nik;
}
```

Method untuk mencari keluarga dimulai dengan mengetahui id_keluarga dari nik penduduk yang wafat, kemudian dipanggil method isBerlaku dalam keluarga dao yang berisi

```
@Override
public void updateIsBerlaku(String id_keluarga){
    String nkk = keluargaMapper.selectNNK(id_keluarga);
    KeluargaModel keluarga = keluargaMapper.selectKeluarga(nkk);
    List<PendudukModel> penduduks = keluarga.getPenduduks();
    int count = 0;
    for (int i=0;i<penduduks.size();i++) {
        if(penduduks.get(i).getIs_wafat() == 1) {
            count++;
        }
    }
    if(count == penduduks.size()){
        keluargaMapper.updateIsBerlaku(nkk);
    }
}
```

Method diatas akan mengiterate semua anggota keluarga dan mengecek jumlah anggota keluarga yang meninggal, jika sudah meninggal semua maka isBerlaku berdasarkan nilai nkk akan merubah atribut istidakberlaku dalam keluarga menjadi true.

8. Fitur Tampilkan Data Penduduk berdasarkan Kota/Kabupaten, Kecamatan, dan Kelurahan tertentu

The screenshot shows a web application titled "SI Kependudukan". The URL in the address bar is "localhost:8080/penduduk/cari". The page has a red header with the title. Below the header, there is a search form with a large input field labeled "Lihat Data Penduduk Berdasarkan Kota". Underneath the input field, there are two dropdown menus: "Kota/Kabupaten" and "Pilih Kota/Kabupaten". At the bottom of the form is a green "LIHAT" button.

Disa Ainun Yolanna

1506689654

APAP A

Fitur ini akan menampilkan penduduk yang berada pada lokasi tertentu yang ditentukan input user.

Contoh Skenario

Pengguna mengakses “/penduduk/cari”



Memilih salah satu kelurahan

Lihat Data Penduduk Berdasarkan Kota

Kota/Kabupaten

KABUPATEN KEPULAUAN SERIBU

LIHAT

Pengguna akan memilih kecamatan berdasarkan kota yang telah dipilih sebelumnya

Disa Ainun Yolanna

1506689654

APAP A

Lihat Data Penduduk Berdasarkan Kecamatan di KABUPATEN KEPULAUAN SERIBU

Kota/Kabupaten

KABUPATEN KEPULAUAN SERIBU



Kecamatan

KEPULAUAN SERIBU UTARA



LIHAT

Pengguna akan memilih kelurahan berdasarkan kota dan kecamatan yang telah dipilih sebelumnya

Lihat Data Penduduk Berdasarkan Kelurahan di KABUPATEN KEPULAUAN SERIBU, Kecamatan KEPULAUAN SERIBU UTARA

Kota/Kabupaten

KABUPATEN KEPULAUAN SERIBU



Kecamatan

KEPULAUAN SERIBU UTARA



Kelurahan

PULAU KELAPA



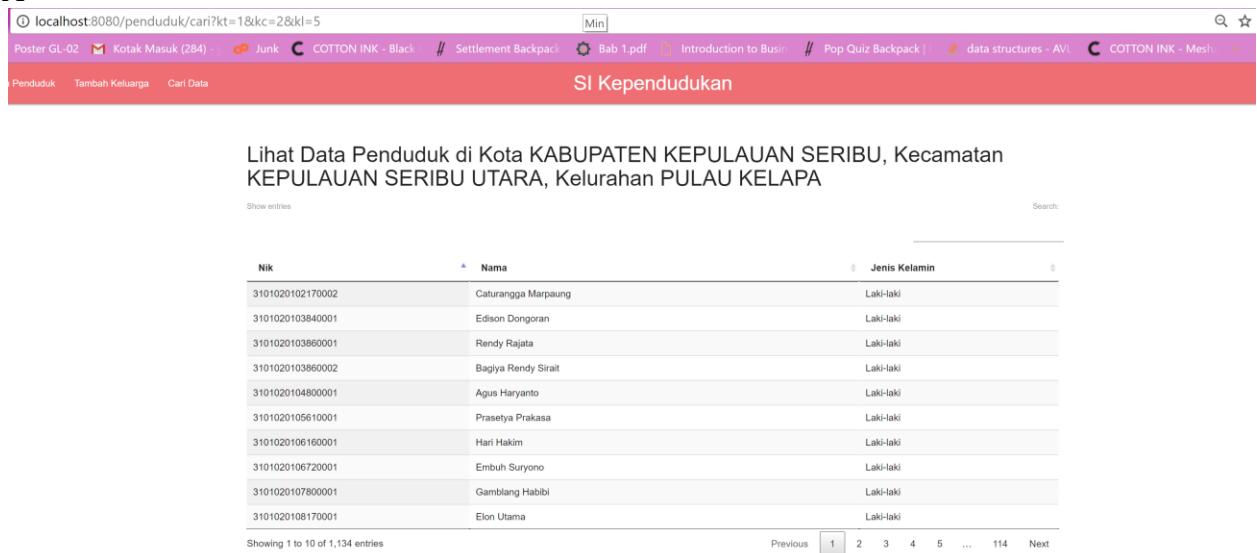
LIHAT

Kemudian setelah mengklik Lihat maka akan ditampilkan penduduk yang terdaftar di lokasi tersebut

Disa Ainun Yolanna

1506689654

APAP A



Nik	Nama	Jenis Kelamin
3101020102170002	Caturangga Marpaung	Laki-laki
3101020103840001	Edision Dongoran	Laki-laki
3101020103860001	Rendy Rajata	Laki-laki
3101020103860002	Bagiya Rendy Sirait	Laki-laki
3101020104800001	Agus Haryanto	Laki-laki
3101020105610001	Prasetya Prakasa	Laki-laki
3101020106160001	Hari Hakim	Laki-laki
3101020106720001	Embuah Suryono	Laki-laki
3101020107800001	Gamblang Habibi	Laki-laki
3101020108170001	Elon Utama	Laki-laki

Langkah pertama saya membuat mapper baru yaitu LokasiMapper yang berisi method-method berikut

```
@Mapper
public interface LokasiMapper {

    @Select("select * from kelurahan where id_kecamatan = #{id_kecamatan}")
    @Results(value = {@Result(property = "kode_kelurahan", column = "id")})
    List<KelurahanModel> selectKelurahans(@Param("id_kecamatan")String id_kecamatan);

    @Select("select * from kecamatan where id_kota = #{id_kota}")
    @Results(value = {@Result(property = "kode_kecamatan", column = "id")})
    List<KecamatanModel> selectKecamatans(@Param("id_kota")String id_kota);

    @Select("select * from kota")
    @Results(value = {
        @Result(property = "kode_kota", column = "id")})
    List<KotaModel> selectKotas();

    @Select("Select nama_kota from kota where id = #{id_kota}")
    String selectNamaKota( String id_kota);

    @Select("Select nama_kecamatan from kecamatan where id = #{id_kecamatan}")
    String selectNamaKecamatan( String id_kecamatan);

    @Select("Select nama_kelurahan from kelurahan where id = #{id_kelurahan}")
    String selectNamaKelurahan( String id_kelurahan);
```

Masing-masing dari method tersebut akan digunakan oleh penduduk controller, method selectKelurahans, selectKecamatans, selectKotas adalah method yang akan menyediakan nilai bagi option yang terdapat pada form-cari-data

Disa Ainun Yolanna

1506689654

APAP A

```
br/>


<form class="input-field" action="/penduduk/cari" method="get">
    <h5 th:if="${flagkota == 'false'}" th:text="Lihat Data Penduduk Berdasarkan Kota"></h5>
    <h5 th:if="${flagkota == 'true' and flagkecamatan == 'false'}" th:text="Lihat Data Penduduk Berdasarkan Kecamatan di ${nama_kota}"></h5>
    <h5 th:if="${flagkota == 'true' and flagkecamatan == 'true'}" th:text="Lihat Data Penduduk Berdasarkan Kelurahan di ${nama_kota} , Kecamatan ${nama_kecamatan}"></h5>

    <div class="row">
        <div class="input-field col s6">
            <div th:if="${flagkota == 'true'}" name="kota" th:value="${id_kota}">
                <select>
                    <option disabled="disabled" selected="selected" th:text="${nama_kota}"></option>
                </select>
                <label >Kota/Kabupaten</label>
                <input hidden="hidden" th:value="${id_kota}" name="kt"/>
            </div>
            <div th:if="${flagkota == 'false'}" name="kota">
                <select name="kt">
                    <option th:value="0" disabled="disabled" selected="selected">Pilih Kota/Kabupaten</option>
                    <div th:each="kota,iterationStatus: ${kotas}">
                        <option th:value="${kota.kode_kota}" th:text="${kota.nama_kota}"></option>
                    </div>
                </select>
                <label >Kota/Kabupaten</label>
            </div>
        </div>
    </div>
    <div class="input-field col s6">
        <div th:if="${flagkecamatan == 'true'}" name="kecamatan" th:value="${id_kecamatan}">
            <select>
                <option disabled="disabled" selected="selected" th:text="${nama_kecamatan}"></option>
            </select>
            <label >Kecamatan</label>
            <input hidden="hidden" th:value="${id_kecamatan}" name="kc"/>
        </div>
    </div>


```

Method selectNamaKelurahan, selectNamaKecamatan, selectNamaKota, akan digunakan untuk memasukan nilai ke option yang sebelumnya telah dipilih. Pada form-cari-data saya menggunakan th:each untuk mengiterate method-method dalam mapper, dan menggunakan banyak flag untuk kondisi yang ditentukan misalnya ketika saya belum memilih kecamatan berarti kelurahan belum bisa dipilih dan optionnya belum bisa ditampilkan.

Singkat cerita saya membuat method di LokasiService, LokasiServiceDatabase sebagai berikut

Disa Ainun Yolanna

1506689654

APAP A

```
public interface LokasiService {  
  
    List<KotaModel> selectKotas();  
  
    List<KecamatanModel> selectKecamatans(String id_kota);  
  
    List<KelurahanModel> selectKelurahans(String id_kecamatan);  
  
    String selectNamaKota(String id_kota);  
  
    String selectNamaKecamatan(String id_kecamatan);  
    String selectNamaKelurahan(String id_kelurahan);  
}  
  
@Service  
public class LokasiServiceDatabase implements LokasiService {  
    @Autowired  
    private LokasiMapper lokasiMapper;  
  
    @Override  
    public List<KotaModel> selectKotas() {  
  
        log.info("select all kota");  
        return lokasiMapper.selectKotas();  
    }  
  
    @Override  
    public List<KecamatanModel> selectKecamatans(String id_kota) {  
        log.info("select all kecamatan");  
        return lokasiMapper.selectKecamatans(id_kota);  
    }  
  
    @Override  
    public List<KelurahanModel> selectKelurahans(String id_kecamatan) {  
        return lokasiMapper.selectKelurahans(id_kecamatan);  
    }  
  
    @Override  
    public String selectNamaKota(String id_kota) { return lokasiMapper.selectNamaKota(id_kota); }  
  
    @Override  
    public String selectNamaKecamatan(String id_kecamatan) { return lokasiMapper.selectNamaKecamatan(id_kecamatan); }  
  
    @Override  
    public String selectNamaKelurahan(String id_kelurahan) { return lokasiMapper.selectNamaKelurahan(id_kelurahan); }  
}
```

Terakhir saya membuat method di PendudukController, yang akan mengecek tiap input user apabila masing-masing input masih null maka penduduk tidak dapat ditemukan, pada fitur ini hal yang pertama harus diisi adalah kota, kemudian kecamatan, dan kelurahan.

Disa Ainun Yolanna

1506689654

APAP A

```
@RequestMapping (value = "/penduduk/cari", method = RequestMethod.GET)
public String cariData(
    @RequestParam(value= "kt", required = false) String id_kota,
    @RequestParam(value= "kc", required = false) String id_kecamatan,
    @RequestParam(value= "kl", required = false) String id_kelurahan,
    Model model){

    List<KotaModel> kotas = lokasiDAO.selectKotas();
    model.addAttribute(attributeName: "flagkota", attributeValue: "false");
    model.addAttribute(attributeName: "kotas", kotas);
    if(id_kota != null) {
        String nama_kota = lokasiDAO.selectNamaKota(id_kota);
        model.addAttribute(attributeName: "flagkota", attributeValue: "true");
        List<KecamatanModel> kecamatans = lokasiDAO.selectKecamatans(id_kota);
        model.addAttribute(attributeName: "id_kota", id_kota);
        model.addAttribute(attributeName: "nama_kota", nama_kota);

        model.addAttribute(attributeName: "flagkecamatan", attributeValue: "false");
        model.addAttribute(attributeName: "kecamatans", kecamatans);

        log.info("id_kecamatan {}", id_kecamatan);
        if(id_kecamatan != null){
            String nama_kecamatan = lokasiDAO.selectNamaKecamatan(id_kecamatan);
            model.addAttribute(attributeName: "flagkota", attributeValue: "true");
            model.addAttribute(attributeName: "flagkecamatan", attributeValue: "true");
            List<KelurahanModel> kelurahans = lokasiDAO.selectKelurahans(id_kecamatan);

            model.addAttribute(attributeName: "id_kota", id_kota);
            model.addAttribute(attributeName: "nama_kota", nama_kota);
            model.addAttribute(attributeName: "id_kecamatan", id_kecamatan);
            model.addAttribute(attributeName: "nama_kecamatan", nama_kecamatan);

            model.addAttribute(attributeName: "flagkelurahan", attributeValue: "false");
            model.addAttribute(attributeName: "kelurahans", kelurahans);

            if(id_kelurahan != null){
                String nama_kelurahan = lokasiDAO.selectNamaKelurahan(id_kelurahan);
                model.addAttribute(attributeName: "nama_kelurahan", nama_kelurahan);
                List<PendudukModel> penduduks = pendudukDAO.selectPenduduksByLocation(id_kota,id_kecamatan,id_kelurahan);
                model.addAttribute(attributeName: "penduduks", penduduks);
            }
        }
    }
    return "view-penduduk-lokasi";
}
```

Jika semua input telah valid dan terisi, maka laman akan mengembalikan “view-penduduk-lokasi” yang dalam html berisi koding mengiterate list of penduduk.

```
<table id="viewall" class="display">
    <thead>
        <tr>
            <th>Nik</th>
            <th>Nama</th>
            <th>Jenis Kelamin</th>
        </tr>
    </thead>
    <tbody>
        <tr th:each="penduduk,iterationStatus: ${penduduks}">
            <td th:text="${penduduk.nik}"></td>
            <td th:text="${penduduk.nama}"></td>
            <td th:text="${(penduduk.jenis_kelamin)==1} ? 'Perempuan' : 'Laki-laki'"></td>
        </tr>
    </tbody>
</table>
```

Optimasi Database

Saya mengoptimasi database dengan menjadikan id di keluarga dan penduduk sebagai primary untuk mempermudah indexing, saya juga melakukan alter column untuk

Disa Ainun Yolanna

1506689654

APAP A

mengincrement apabila terdapat penduduk baru atau keluarga baru untuk meringkankan program saya.