Write up tp apap 1 :

1. Fitur 1

Fitur ini cuma ngemap dari database ke tampilan. Ada yang perlu diubah formatnya, spt jenis kelamin. Ini bisa pake method spt di bawah :

```
public String jenisKelamin() {
    return jenis_kelamin == 0 ? "Pria" : "Wanita";
}
```

2. Fitur 2

Masukkan 2 attribute ke view, keluarga, dan penduduks yang merupakan bagian dr keluarga itu. Pertama fetch dulu keluarga nya berdasarkan nkk, terus fetch penduduks berdasarkan id keluarga tsb, yang disimpet dalam field.

```
List<PendudukModel> penduduks = pendudukDAO.selectPendudukviaIDKeluarga(keluarga.getId());
model.addAttribute("penduduks", penduduks);
model.addAttribute("keluarga", keluarga);
return "keluarga-info";
```

3. Fitur 3

Cuma ngambil dari kode kecamatan. Pertama fetch dulu penduduk dari NIK, yang bisa dapet id_keluarga, dst sampe kecamatan. Potong satu karakter paling terakhir.

Untuk, tanggal, bisa di format sendiri pake koding biasa (split, substring, dll)

```
String[] tanggals = tanggal_lahir.split("-");
if (jenis_kelamin == 1) {
    tanggals[2] = (new Integer(tanggals[2]) + 40) + "";
}

StringBuilder nik = new StringBuilder();
nik.append(kecamatan.getKode_kecamatan().substring(0, 6));
// String str = nik.toString();
nik.append(tanggals[2]).append(tanggals[1]).append(new Integer(tanggals[0]) % 100);
```

Untuk urutan index, juga di koding biasa lewat loop dan mod

```
for (int i = 1; i < 10000; i++) {
    StringBuilder testedNIK = new StringBuilder().append(nik.toString());
    if (i < 10) {
        testedNIK.append("000" + i);
    } else if (i < 100) { // i>=10
        testedNIK.append("00" + i);
    } else if (i < 1000) { // i >= 100
        testedNIK.append("0" + i);
    } else { // i >= 1000
        testedNIK.append(i);
    }
    PendudukModel pm = pendudukDAO.quickNIKCheck(testedNIK.toString());
    if (pm == null) {
        nik = testedNIK;
        break;
    }
```

4. Fitur 4

Saya tidak menghandle apakah kota tsb emg ada kelurahan dan kecamatan yang ada di dalamnya. Jadi lebih baik, pake id kelurahan saja. Dari situ, pake saja approach yang sama spt NIK, cuman tanggal nya di ganti today.

5. Fitur 5

Masalah dari fitur ini adalah, saya belum paham cara ngeset secara konditional untuk dropdown. Karena permintaan cuma ttg terisi, jadi saya set secara random dan sama semua. Sisanya, field bebas saya beri nilai aja sesuai dengan apa yang ada di database.

```
<form th:action="'/penduduk/ubah/' + ${nik}" method="post" th:object="${penduduk}">
    <br/><label for="nama">Nama</label> <input type="text" name="nama" th:field="*{nama}" th:value=
    "${penduduk.nama}"/>
    <br/><label for="tempat_lahir">Tempat Lahir</label> <input type="text" name="tempat_lahir" th:field=
    "*{tempat_lahir}" th:value="${penduduk.tempat_lahir}"  />
    <br/><label for="tanggal_lahir'">Tanggal Lahir</label> <input type="date" name="tanggal_lahir" th:field=
    "*{tanggal_lahir}" th:value="${penduduk.tanggal_lahir}"  />
    <br/><label for="golongan_darah">Golongan Darah</label>
        <select name="golongan_darah" >
            <option selected="selected" value="A">A</option>
            <option value="B">B</option>
            <option value="AB">AB</option>
            <option value="O">O</option>
        </select>
    <br/><label for="agama">Agama</label> <input type="text" name="agama" th:field="*{agama}" th:value=
    "${penduduk.agama}"  />
    <br/><label for="status_perkawinan">Status Perkawinan</label>
        <select name="status_perkawinan">
            <option value="Kawin">Kawin</option>
            <option selected="selected" value="Belum Kawin">Belum Kawin</option>
            <option value="Cerai Mati">Cerai Mati</option>
            <option value="Cerai Hidup">Cerai Hidup</option>
        </select>
    <br/><label for="pekerjaan">Pekerjaan</label> <input type="text" name="pekerjaan"  th:field="*{pekerjaan}"
```

Jadi pertama, kita fetch dulu penduduk dari NIK nya. Dari situ, semua data bisa di fetch dan dimasukkan ke dalam form. User bisa ganti-ganti itu, tapi ya ga harus.

```
PendudukModel pendudukOld = pendudukDAO.selectPendudukViaNIK(nik);
if (flag == null) {
    model.addAttribute("penduduk", pendudukOld);
    model.addAttribute("nik", nik);
    return "form-ubah-penduduk";

}
```

Untuk ngebedain apakah dia mengakses page itu baru mau ngisi atau sudah submit, dikasih flag pada form (typenya hidden) sehingga, methodnya selalu request param meskipun pas baru mau ngubah datanya, semua data null. Nah, karena semua data null (termasuk flagnya,) program bisa ngebedain apakah mau nyimpen apa mau ngisi.

```
@RequestMapping("/penduduk/ubah/{nik}")
public String ubahPendudukviaNIK(Model model, @PathVariable(value = "nik") String nik,
        @RequestParam(value = "flag", required = false) String flag,
        @RequestParam(value = "nama", required = false) String nama,
        @RequestParam(value = "tempat_lahir", required = false) String tempat_lahir,
        @RequestParam(value = "tanggal_lahir", required = false) String tanggal_lahir,
        @RequestParam(value = "status_perkawinan", required = false) String status_perkawinan,
        @RequestParam(value = "agama", required = false) String agama,
        @RequestParam(value = "jenis_kelamin", required = false) Integer jenis_kelamin,
        @RequestParam(value = "golongan_darah", required = false) String golongan_darah,
        @RequestParam(value = "pekerjaan", required = false) String pekerjaan,
        @RequestParam(value = "kewarganegaraan", required = false) Integer is_wni,
        @RequestParam(value = "id_keluarga", required = false) Integer id_keluarga,
        @RequestParam(value = "status_kematian", required = false) Integer is_wafat,
        @RequestParam(value = "status_dalam_keluarga", required = false) String status_dalam_keluarga)
```

In case NIK nya ganti, pertama di fetch dulu semua data yang lama. Lalu, kalo user udh nginput semua data, buat model baru dengan semua input, tapi pakai NIK yang baru. Panggil update setelah itu untuk mapping ke database.

```
PendudukModel pendudukOld = pendudukDAO.selectPendudukViaNIK(nik);
if (flag == null) {
    model.addAttribute("penduduk", pendudukOld);
    model.addAttribute("nik", nik);
    return "form-ubah-penduduk";

}

String newNIK = generateNIKFrom(id_keluarga, jenis_kelamin, tanggal_lahir);

PendudukModel pendudukNew = new PendudukModel(pendudukOld.getId(), newNIK, nama, tempat_lahir,
        id_keluarga, agama, pekerjaan, status_perkawinan, status_dalam_keluarga, golongan_darah
        is_wafat);

pendudukDAO.updatePenduduk(pendudukNew);
model.addAttribute("nik", nik);

return "success-update";
```

Perlu diperhatikan, program ini menganggap IDnya udh auto increment. Jadi, memang nggak masuk ke SQL. kalo IDnya nggak auto increment, ada kemungkinan gagal programnya

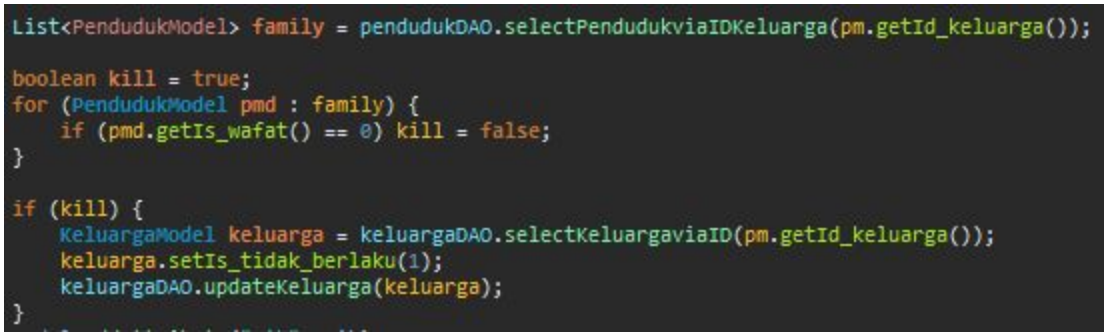
6. Fitur 6

Approachnya sama dengan fitur 5.

7. Fitur 7

Cuma fetch data dari NIK, terus set is_wafatnya jadi 1, lalu update.

Untuk masalah kematian kartu keluarga, pertama disearch dulu apakah masih ada keluarga yang hidup. Kalo masih ada, jangan di set non-aktif.



```
List<PendudukModel> family = pendudukDAO.selectPendudukviaIDKeluarga(pm.getId_keluarga());

boolean kill = true;
for (PendudukModel pmd : family) {
    if (pmd.getIs_wafat() == 0) kill = false;
}

if (kill) {
    KeluargaModel keluarga = keluargaDAO.selectKeluargaviaID(pm.getId_keluarga());
    keluarga.setIs_tidak_berlaku(1);
    keluargaDAO.updateKeluarga(keluarga);
}
```