Penjelasan MVC

Sebelum memulai mengerjakan tugas, saya terlebih dahulu melakukan perbaikan pada application properties agar database terhubung dengan tabel pada siduk.sql(tugas1). Setelah itu membuat model untuk semua tabel beserta variable dan paramaternya.

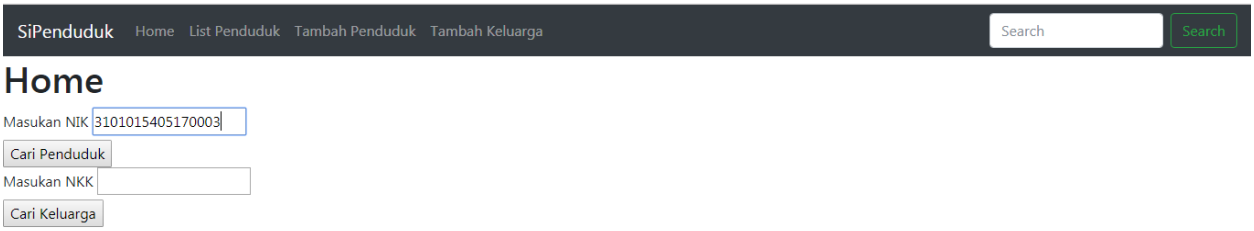
1. Tampilkan Data Penduduk Berdasarkan Nik
   Untuk Mengerjakan tugas ini yang saya lakukan adalah membuat mapping sesuai dengan permintaan soal. Kemudian membuat mapper select untuk mengambil data penduduk melalui nik.

```
@Select("select id, nik, nama, status_dalam_keluarga, tempat_lahir, id_keluarga, tanggal_lahir, jenis_kelamin, golongan_darah, agama,
"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 = "jenis_kelamin", column = "jenis_kelamin"), @Result(property = "golongan_darah", column = "golongan_darah")
        @Result(property = "agama", column = "agama"), @Result(property = "is_wni", column = "is_wni"),
        @Result(property = "status_perkawinan", column = "status_perkawinan"),
        @Result(property = "id", column = "id"),
        @Result(property = "status_dalam_keluarga", column = "status_dalam_keluarga"),
        @Result(property = "pekerjaan", column = "pekerjaan"), @Result(property = "is_wafat", column = "is_wafat"),
        @Result(property = "id_keluarga", column = "id_keluarga") })
PendudukModel selectpenduduk(@Param("nik") String nik);

@Select("select id,id_kelurahan, alamat, rt, rw " +
"from keluarga where id = #{id_keluarga}")
@Results(value = { @Result(property = "alamat", column = "alamat"),
        @Result(property = "rt", column = "rt"),
        @Result(property = "rw", column = "rw")})
KeluargaModel selectkeluarga(@Param("id_keluarga") String id_keluarga);

@Select("select * " +
"from kecamatan where id = #{id_kecamatan}")
@Results(value = { @Result(property = "nama_kecamatan", column = "nama_kecamatan"),
                @Result(property = "id", column = "id"),})
KecamatanModel selectkecamatan(@Param("id_kecamatan") String id_kecamatan);

@Select("select id,nama_kota " +
"from kota where id = #{id_kota}")
@Results(value = { @Result(property = "nama_kota", column = "nama_kota")})
KotaModel selectkota(@Param("id_kota") String id_kota);
```

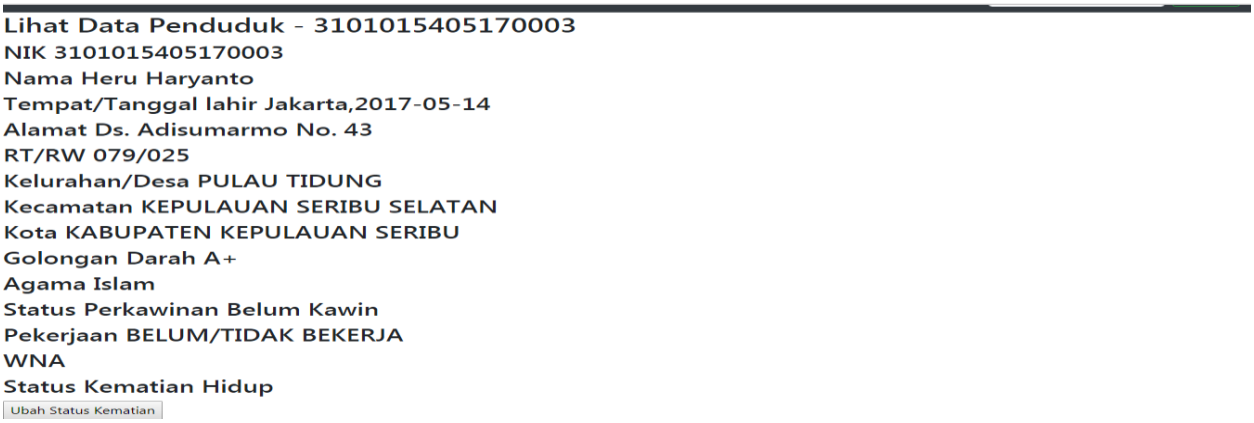
Ini adalah mapper yang digunakan untuk mengerjakan soal no 1. Kemudian baru saya membuat service database dimana berisi logic untuk menghubungkan tabel. Ini adalah fungsi dari servicedatabase yang digunakan untuk mengerjakan soal 1. Setelah itu saya membuat service untuk menentukan fungsi apa saja yang diperlukan oleh servicedatabase

```
//soal 1
@Override
public PendudukModel selectpenduduk (String nik) {
    PendudukModel penduduk = pendudukMapper.selectpenduduk(nik);
    KeluargaModel keluarga = pendudukMapper.selectkeluarga(penduduk.getId_keluarga());
    KelurahanModel kelurahan = pendudukMapper.selectkelurahan(keluarga.getId_kelurahan());
    KecamatanModel kecamatan = pendudukMapper.selectkecamatan(kelurahan.getId_kecamatan());
    KotaModel kota = pendudukMapper.selectkota(kecamatan.getId_kota());
    penduduk.setKeluarga(keluarga);
    keluarga.setKelurahan(kelurahan);
    kelurahan.setKecamatan(kecamatan);
    kecamatan.setKota(kota);
    return penduduk;
}
```

Baru setelah itu membuat tampilan, tampilan yang digunakan hanya sekedar untuk mengetahui bahwa fungsi berjalan atau tidak.



Gambar diatas adalah fitur setelah menerapkan bootstrap dengan cara fragmentasi. Setelah cari penduduk maka akan keluar tampilan sebagai berikut.



Berikut adalah fragments serta index html, dan viewPenduduk yang digunakan untuk soal 1

```
    <head>
        <title>View Penduduk by NIK</title>
        <link rel="stylesheet" href="/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://cdn.datatables.net/v/dt/dt-1.10.16/datatables.min.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdn.datatables.net/1.10.7/js/jquery.dataTables.min.js"></script>
<script type="text/javascript" src="/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#table').DataTable();
    });
</script>
    </head>
    <body>
    <div th:replace="fragments/fragment :: header"></div>
     <h2 th:text="'Lihat Data Penduduk - ' + ${penduduk.nik}"></h2>
        <h3 th:text="'NIK ' + ${penduduk.nik}">Penduduk NIK</h3>
        <h3 th:text="'Nama ' + ${penduduk.nama}">Penduduk Name</h3>
        <h3 th:text="'Tempat/Tanggal lahir ' + ${penduduk.tempat_lahir} + ',' + ${penduduk.tanggal_lahir}">Penduduk BirthDate</h3>
        <h3 th:text="'Alamat ' + ${penduduk.keluarga.getAlamat()}">Penduduk Alamat</h3>
        <h3 th:text="'RT/RW ' + ${penduduk.keluarga.getRt()} + '/' + ${penduduk.keluarga.getRw()}">Penduduk Religion</h3>
        <h3 th:text="'Kelurahan/Desa ' + ${penduduk.keluarga.kelurahan.getNama_kelurahan()}">Penduduk Religion</h3>
        <h3 th:text="'Kecamatan ' + ${penduduk.keluarga.kelurahan.kecamatan.getNama_kecamatan()}">Penduduk Religion</h3>
        <h3 th:text="'Kota ' + ${penduduk.keluarga.kelurahan.kecamatan.kota.getNama_kota()}">Penduduk Religion</h3>
        <h3 th:text="'Golongan Darah ' + ${penduduk.golongan_darah}">Penduduk Blood Type</h3>
        <h3 th:text="'Agama ' + ${penduduk.agama}">Penduduk Religion</h3>
        <h3 th:text="'Status Perkawinan ' + ${penduduk.status_perkawinan}">Penduduk Marriage Status</h3>
        <h3 th:text="'Pekerjaan ' + ${penduduk.pekerjaan}">Penduduk Job</h3>
        <h3 th:text="'Kewarganegaraan ' + ${penduduk.is_wni} == 1 ? 'WNI' : 'WNA'">Penduduk Kewarganegaraan</h3>
        <h3 th:text="'Status Kematian ' + ${penduduk.is_wafat == 0 ? 'Hidup' : 'Mati'}">Penduduk Alive</h3>

    <form action="/penduduk/mati" method="post" >
    <input type="hidden" name="nik" th:value="${penduduk.nik}" />
        <div>
            <button type="submit">Ubah Status Kematian</button>
        </div>
    </form>

```

Index html

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>

<title>Home
</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" href="/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://cdn.datatables.net/v/dt/dt-1.10.16/datatables.min.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdn.datatables.net/1.10.7/js/jquery.dataTables.min.js"></script>
<script type="text/javascript" src="/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#table').DataTable();
    });
</script>
</head>

<body>
    <div th:replace="fragments/fragment :: header"></div>
    <h1 class="page-header">Home</h1>

    <form action="/penduduk" method="get">
        <div>
            <label for="nik">Masukan NIK</label> <input type="text" name="nik" />
        </div>
        <div>
            <button type="submit">Cari Penduduk</button>
        </div>
    </form>

    <form action="/keluarga" method="get">
        <div>
            <label for="nik">Masukan NKK</label> <input type="text" name="nkk" />
        </div>
        <div>
```

Fragments

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
</head>
<body>
<div th:fragment="header">
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
<a class="navbar-brand" href="/penduduk/cari">SiPenduduk</a>
<button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
<span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarSupportedContent">
<ul class="navbar-nav mr-auto">
<li class="nav-item">
<a class="nav-link" href="/">Home</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/penduduk/cari">List Penduduk</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/penduduk/tambah">Tambah Penduduk</a>
</li>
<li class="nav-item">
<a class="nav-link" href="/keluarga/tambah">Tambah Keluarga</a>
</li>
</ul>
<form class="form-inline my-2 my-lg-0">
<input class="form-control mr-sm-2"
type="text"
placeholder="Search"
aria-label="Search" />
<button class="btn btn-outline-success my-2 my-sm-0" type="submit">
Search
</button>
</form>
```

2. Tampilkan Daftar Penduduk Berdasarkan Nkk

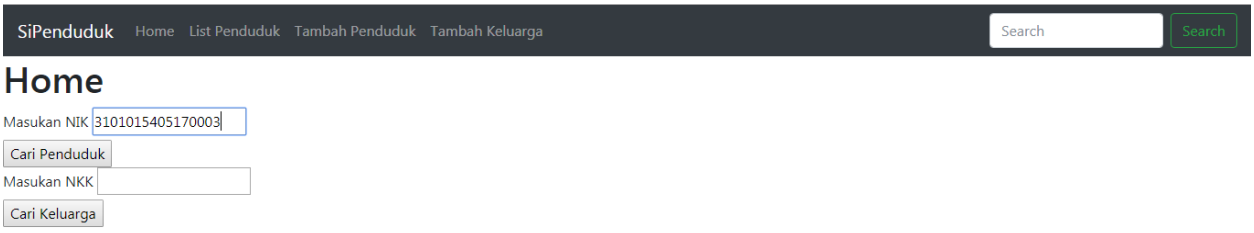
   Untuk Mengerjakan soal ini yang saya lakukan adalah membuat mapping sesuai dengan permintaan soal. Kemudian membuat mapper select untuk mengambil data penduduk melalui nkk. Dalam hal ini saya membutuhkan 2 mapper tambahan yang memiliki fungsi untuk mencari informasi id keluarga berdasarkan nkk. Dan mendapatkan list penduduk dengan id keluarga. Ini adalah mapper yang digunakan untuk mengerjakan soal no 2.

```
//Soal 2
@Select("select nik, nama, tempat_lahir, id_keluarga, tanggal_lahir, jenis_kelamin, golongan_darah, agama, is_wni, status_perkawinan,
"from penduduk where id_keluarga = #{id}")
@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 = "is_wni", column = "is_wni"),
        @Result(property = "status_perkawinan", column = "status_perkawinan"),
        @Result(property = "perkerjaan", column = "perkerjaan"), @Result(property = "is_wafat", column = "is_wafat"),
        @Result(property = "id_keluarga", column = "id_keluarga") })
List<PendudukModel> selectpenduduks(@Param("id") String id);

@Select("select id, nomor_kk, id_kelurahan, alamat, rt, rw " +
"from keluarga where nomor_kk = #{nkk}")
@Results(value = { @Result(property = "alamat", column = "alamat"),
        @Result(property = "id", column = "id"),
        @Result(property = "nomor_kk", column = "nomor_kk"),
        @Result(property = "rt", column = "rt"),
        @Result(property = "rw", column = "rw")})
KeluargaModel selectkeluarga2(@Param("nkk") String nkk);
```

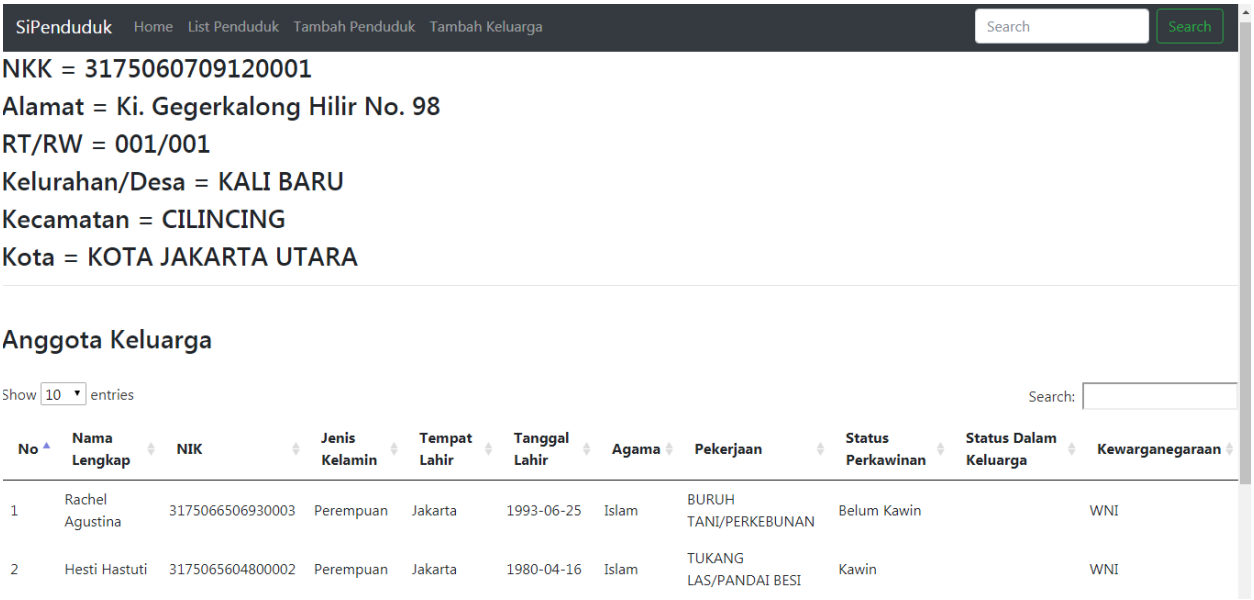
Setelah itu saya menambahakn service untuk membuat *interface* yang akan dipakai pada soal 2 Kemudian baru saya membuat service database dimana berisi logic untuk menghubungkan tabel. Ini adalah fungsi dari servicedatabase yang digunakan untuk mengerjakan soal 2. Ada beberapa fungsi dari mapper yang sudah dibuat pada soal no 1 yang bisa diguanakn. Berikut adalah *servicedatabase* yang dibuat

```
//soal 2
@Override
public KeluargaModel selectkeluarga (String nkk) {
    KeluargaModel keluarga = pendudukMapper.selectkeluarga2(nkk);
    List<PendudukModel> penduduk = pendudukMapper.selectpenduduks(keluarga.getId());
    KelurahanModel kelurahan = pendudukMapper.selectkelurahan(keluarga.getId_kelurahan());
    KecamatanModel kecamatan = pendudukMapper.selectkecamatan(kelurahan.getId_kecamatan())
    KotaModel kota = pendudukMapper.selectkota(kecamatan.getId_kota());
    keluarga.setPenduduks(penduduk);
    keluarga.setKelurahan(kelurahan);
    kelurahan.setKecamatan(kecamatan);
    kecamatan.setKota(kota);
    return keluarga;
}
```

Baru setelah itu membuat tampilan, tampilan yang digunakan hanya sekedar untuk mengetahui bahwa fungsi berjalan atau tidak.



Gambar diatas adalah fitur setelah menerapkan bootstrap dengan cara fragmentasi. Setelah cari penduduk maka akan keluar tampilan sebagai berikut.



Contoh potongan file html yang ditambahkan pada soal ini

```
</script>
    </head>
    <body>
    <div th:replace="fragments/fragment :: header"></div>
        <h3 th:text="'NKK = ' + ${keluarga.nomor_kk}">NKK KELUARGA</h3>
        <h3 th:text="'Alamat = ' + ${keluarga.alamat}">ALAMAT KELUARGA</h3>
        <h3 th:text="'RT/RW = ' + ${keluarga.rt} + '/' + ${keluarga.rt}">RT/RW</h3>
    <h3 th:text="'Kelurahan/Desa =  ' + ${keluarga.kelurahan.getNama_kelurahan()}">KELURAHAN</h3>
        <h3 th:text="'Kecamatan = ' + ${keluarga.kelurahan.kecamatan.getNama_kecamatan()}">KECAMATAN</h3>
        <h3 th:text="'Kota = ' + ${keluarga.kelurahan.kecamatan.kota.getNama_kota()}">KOTA</h3>
    <hr/>
    <br/>
    <h3>Anggota Keluarga</h3>
    <br/>

    <table id = "table">
   <thead>
     <tr>
    <th>No</th>
    <th>Nama Lengkap</th>
                <th>NIK</th>
    <th>Jenis Kelamin</th>
    <th>Tempat Lahir</th>
    <th>Tanggal Lahir</th>
    <th>Agama</th>
    <th>Pekerjaan</th>
    <th>Status Perkawinan</th>
    <th>Status Dalam Keluarga</th>
    <th>Kewarganegaraan</th>
    </tr>
   </thead>
   <tbody>
       <tr th:each="penduduk,iterationStatus: ${keluarga.penduduks}">
        <td th:text="${iterationStatus.count}">No. 1</td>
        <td th:text="${penduduk.nama}">Nama Penduduk</td>
```

3. Fitur Add Penduduk dan
4. Fitur Add Keluarga

Sama halnya dengan cara pengerjaan no 1 dan no 2, yang dilakukan adalah membuat menambahkan mapper sesuai dengan kebutuhan dalam hal ini saya menggunakan *insert*

```
// soal 3
@Select("Select max(id) from penduduk")
String getMaxId();

//soal 4
@Select("Select max(id) from keluarga")
String getMaxIdKeluarga();

//soal 3
@Select("Select nik from penduduk where nik between #{min} and #{max} and jenis_kelamin = #{jenis_kelamin} Order by nik desc limit 1")
String getNikPenduduk(@Param("min") String min, @Param("max") String max, @Param("jenis_kelamin") String jenis_kelamin);

@Insert("insert into penduduk(id,nik,nama,tempat_lahir,tanggal_lahir,jenis_kelamin, is_wni, id_keluarga,agama,pekerjaan,status_perkawi
"#{agama}, #{pekerjaan}, #{status_perkawinan}, #{status_dalam_keluarga}, #{golongan_darah}, #{is_wafat})")
void addpenduduks(PendudukModel penduduk);

// soal 4
@Insert("insert into keluarga(id,nomor_kk,alamat,rt,rw,id_kelurahan,is_tidak_berlaku) values (#{id}, #{nomor_kk}, #{alamat}, #{rt}, #{
    "#{is_tidak_berlaku}) ")
void addkeluargas(KeluargaModel keluarga);
```

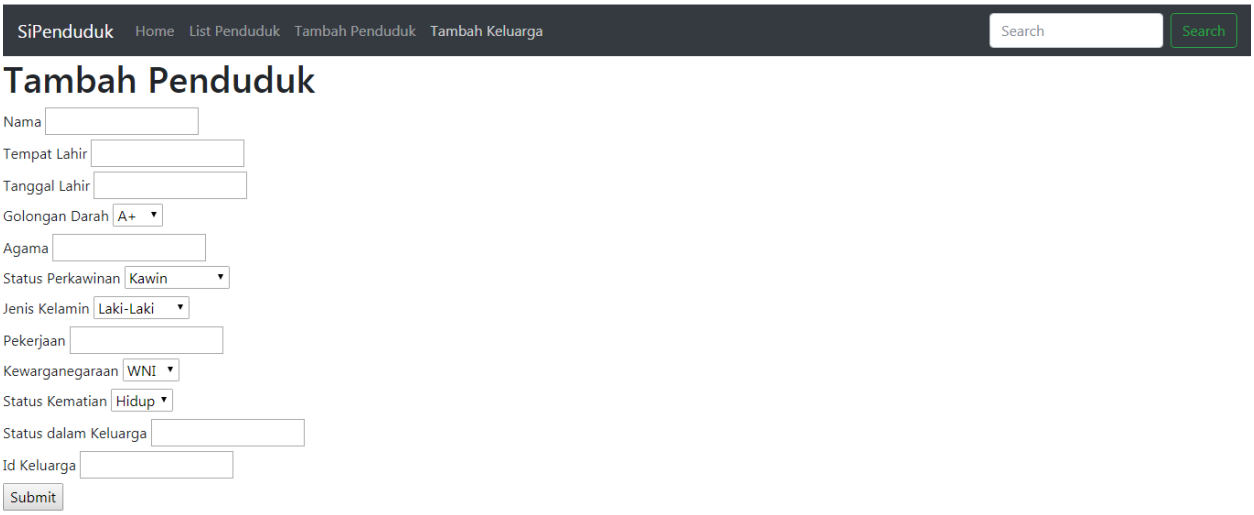
Fungsi baru yang digunakan adalah get max id, untuk mendapatkan id paling tinggi, digunakan untuk generate id dan fungsi untuk mencari 12 string awal yang sama.

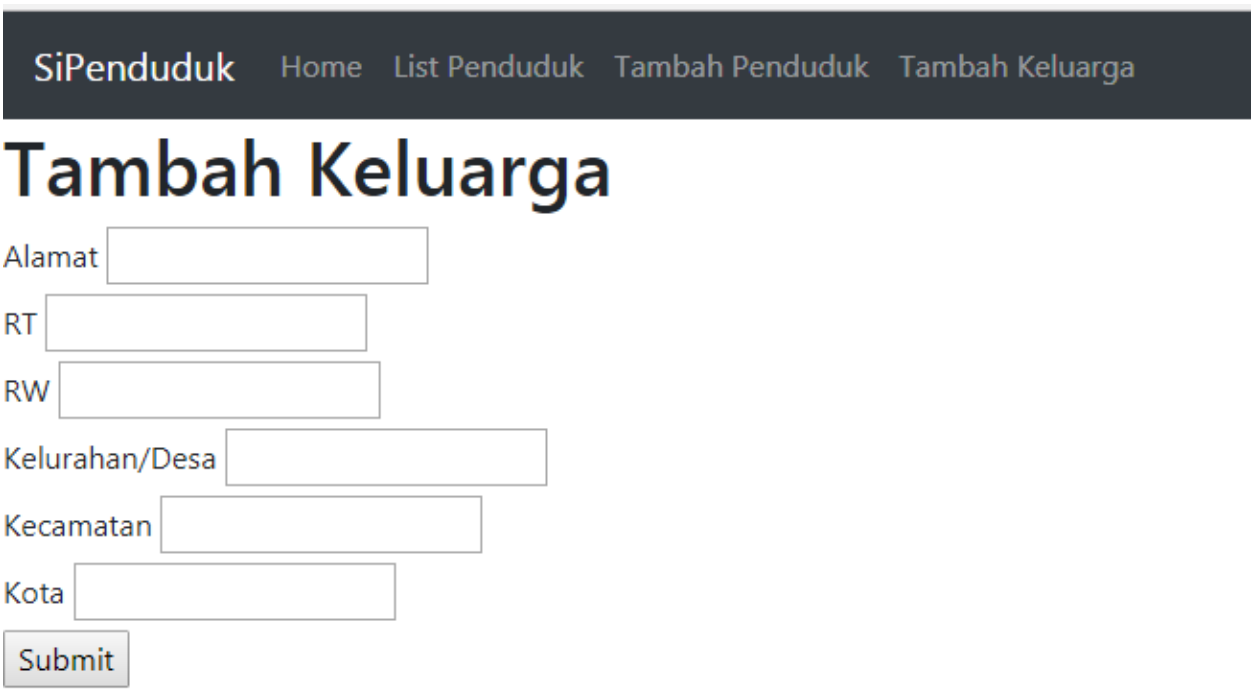
Kemudian menambahkan method pada *servicedatabase* disini juga terjadi perhitungan construct nik sesuai permintaan soal. Method lain yang ditambahkan adalah add penduduk pada service database yang digunakan untuk melakukan insert ke database. Pada soal 3 dan soal 4 ini beberapa fungsi di mapper yang telah dibuat sebelumnya juga bisa diterapkan.

```
    //soal 3
    @Override
    public PendudukModel addpenduduk (String nama, String tempat, String tanggal, String is_wni, String pekerjaan, String golongan_darah,
        KeluargaModel keluarga = pendudukMapper.selectkeluarga(id_keluarga);
        KelurahanModel kelurahan = pendudukMapper.selectkelurahan(keluarga.getId_kelurahan());
        KecamatanModel kecamatan = pendudukMapper.selectkecamatan(kelurahan.getId_kecamatan());
        String nik1 = kecamatan.getKode_kecamatan();
        String[] tanggals = tanggal.split("-");
        String tanggalf = tanggals[2] + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
        System.out.println(tanggalf);
        nik1 = nik1.substring(0, nik1.length() - 1);
        if(jenis_kelamin.equals("1")){
            int nik2 = Integer.parseInt(tanggals[2].substring(0,1));
            nik2 += 4;
            System.out.println(nik2);
            String temp =nik2 + tanggals[2].substring(1,2);
            tanggalf = temp + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
            System.out.println(tanggalf);
        }
        String nik = nik1 + tanggalf;
        String min = nik +"0001";
        String max = nik +"0999";
        String nik4 = pendudukMapper.getNikPenduduk(min,max, jenis_kelamin);
        String nikf = "";
        if(nik4 != null){
        nik4 = nik4 = nik4.substring(12, nik4.length());
        int yes = Integer.parseInt(nik4) +1;
            if (yes < 10 ){
            nikf = nik + "000" + yes;
            }
            else if (yes < 100 ){
            nikf = nik + "00"+ yes;
            }
            else if (yes < 1000 ){
            nikf = nik + "0"+ yes;
            }
        }
```

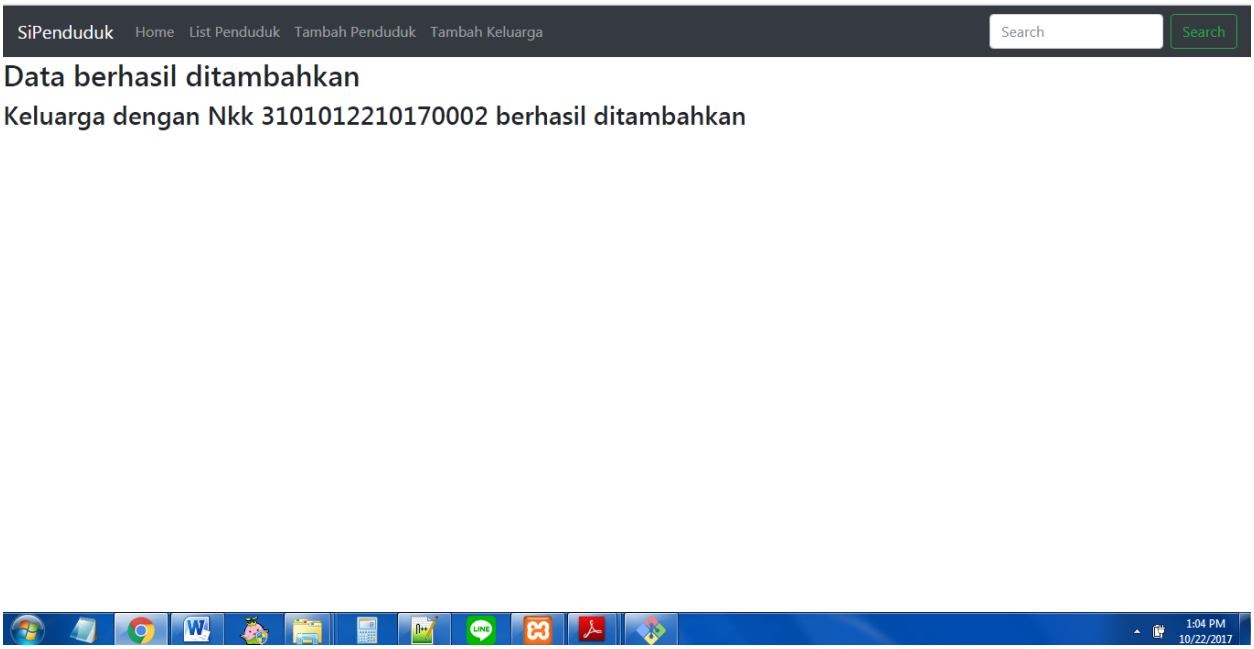
```java
//soal 4
@Override
public KeluargaModel addKeluarga(String alamat, String rt, String rw, String nama_kelurahan){
    String awal6 = pendudukMapper.selectkodekelurahan(nama_kelurahan);
    String id_kelurahan = pendudukMapper.selectidkelurahan(nama_kelurahan);
        System.out.println(id_kelurahan);
    awal6 = awal6.substring(0,6);
        System.out.println(awal6);
    String date = new SimpleDateFormat("yyyy-MM-dd").format(new Date());
    System.out.println(date);
    String[] tanggals = date.split("-");
    String tanggalf = tanggals[2] + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
    String nkk = awal6 + tanggalf;
    String min = nkk +"0001";
    System.out.println(nkk);
    String max = nkk +"0999";
    System.out.println(max);
    String nkkf = pendudukMapper.getNkkKeluarga(min,max);
    if(nkkf != null){
    System.out.println(nkkf);
    nkkf = nkkf = nkkf.substring(12, nkkf.length());
    int yes = Integer.parseInt(nkkf) +1;
        if (yes < 10 ){
        nkkf =nkk + "000"+ yes;
        }
        else if (yes < 100 ){
        nkkf =nkk + "00"+ yes;
        }
        else if (yes < 1000 ){
        nkkf =nkk + "0"+ yes;
        }
    }
    else{
        nkkf = nkk + "0001";
```

Dan tampilan





Setelah berhasil add maka akan mucul tampilan



5. Fitur 5 dan 6 update

Pada mapper ditambahkan

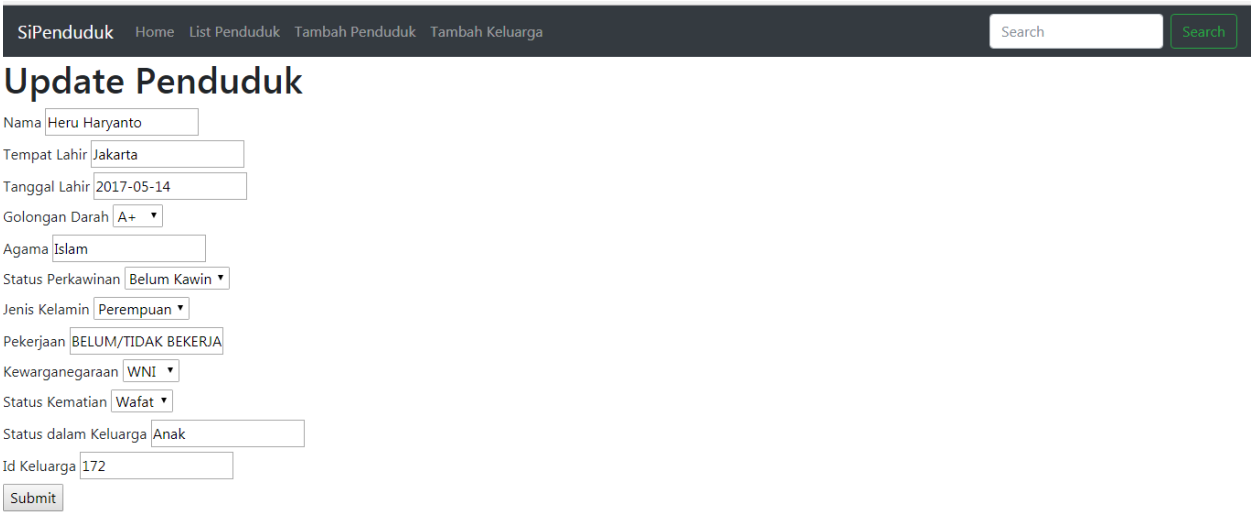
```
// soal 5
@Update("update penduduk SET nik = #{nik}, nama = #{nama}, tempat_lahir = #{tem
"agama = #{agama}, pekerjaan = #{pekerjaan}, status_perkawinan = #{status_perka
void updatePenduduk(PendudukModel penduduk);

//soal 6
@Update("update keluarga SET nomor_kk = #{nomor_kk}, alamat = #{alamat}, RT = #
void updateKeluarga(KeluargaModel keluarga);
```

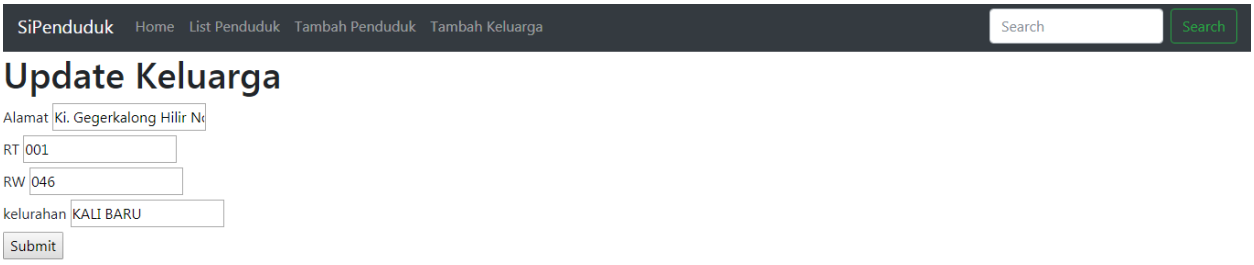
Pada Service Database

```
//soal 5
@Override
public PendudukModel uppenduduk (String id, String nama, String tempat, String tanggal, String is_wni, String pekerjaan, Strin
    KeluargaModel keluarga = pendudukMapper.selectkeluarga(id_keluarga);
    KelurahanModel kelurahan = pendudukMapper.selectkelurahan(keluarga.getId_kelurahan());
    KecamatanModel kecamatan = pendudukMapper.selectkecamatan(kelurahan.getId_kecamatan());
    String nik1 = kecamatan.getKode_kecamatan();
    String[] tanggals = tanggal.split("-");
    String tanggalf = tanggals[2] + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
    System.out.println(tanggalf);
    nik1 = nik1.substring(0, nik1.length() - 1);
    if(jenis_kelamin.equals("1")){
        int nik2 = Integer.parseInt(tanggals[2].substring(0,1));
        nik2 += 4;
        System.out.println(nik2);
        String temp =nik2 + tanggals[2].substring(1,2);
        tanggalf = temp + tanggals[1] + tanggals[0].substring(2,tanggals[0].length());
        System.out.println(tanggalf);
    }
    String nik = nik1 + tanggalf;
    String min = nik +"0001";
    String max = nik +"0999";
    String nik4 = pendudukMapper.getNikPenduduk(min,max, jenis_kelamin);
    String nikf = "";
    if(nik4 != null){
    nik4 = nik4 = nik4.substring(12, nik4.length());
    int yes = Integer.parseInt(nik4) +1;
        if (yes < 10 ){
        nikf = nik + "000" + yes;
        }
        else if (yes < 100 ){
        nikf = nik + "00"+ yes;
        }
        else if (yes < 1000 ){
        nikf = nik + "0"+ yes;
        }
    }
```

Sama halnya saat membuat penduduk dan keluarga baru hal yang berbeda adalah tidak perlu membuat id baru.

Untuk menyelesaikan soal ini diperlukan 2 mapping mapping /penduduk/update dan penduduk/update/submit. Sama halnya dengan soal 6 memerlukan 2 mapping. Mapping penduduk/update pertama digunakan untuk menampilkan form yang sudah diisi sesuai dengan database berikut hasil tampilan dan code





Source Code

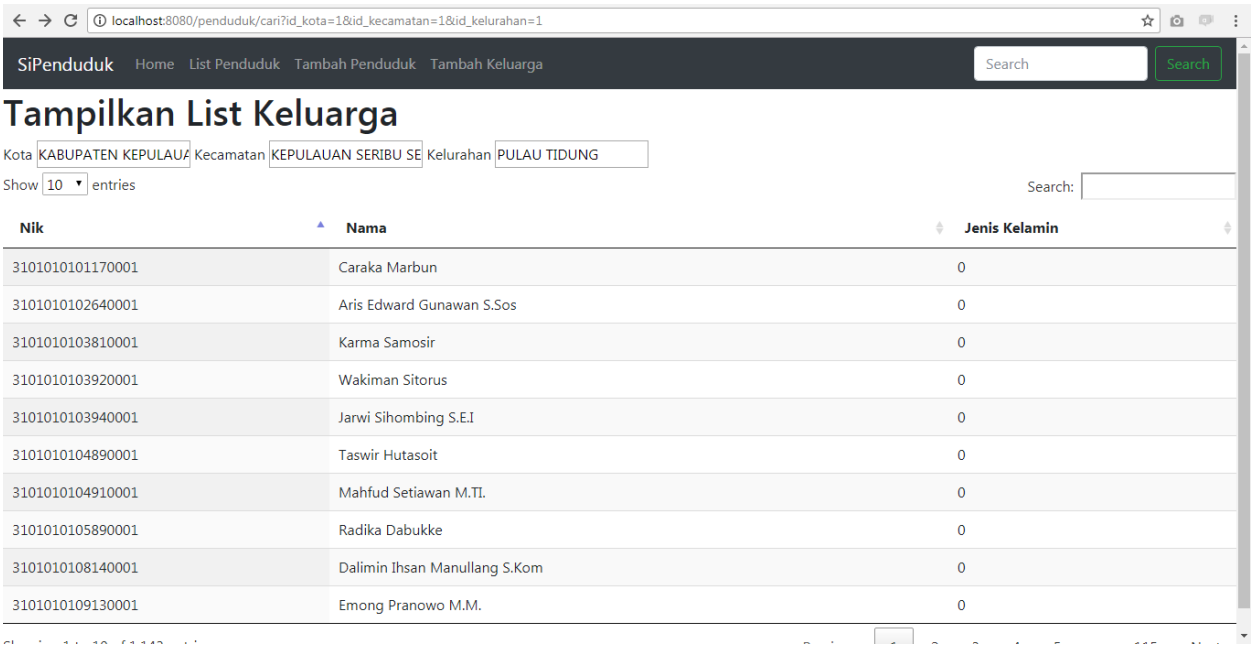
```
    xmlns:th="http://www.thymeleaf.org">
<head>

<title>Update Keluarga</title>
<link rel="stylesheet" href="/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://cdn.datatables.net/v/dt/dt-1.10.16/datatables.min.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdn.datatables.net/1.10.7/js/jquery.dataTables.min.js"></script>
<script type="text/javascript" src="/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#table').DataTable();
    });
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>
    <div th:replace="fragments/fragment :: header"></div>
    <h1 class="page-header">Update Keluarga</h1>

    <form action="/keluarga/update/submit" method="post" >
        <input type="hidden" name="nkk" th:value="${keluarga.nomor_kk}" />
        <div>
            <label for="nama">Alamat</label> <input type="text" name="nama" th:value="${keluarga.
        </div>
        <div>
            <label for="rt">RT</label> <input type="text" name="rt" th:value="${keluarga.rt}" th:
```

```
<!DOCTYPE HTML>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org">
<head>

<title>Update Penduduk</title>
<link rel="stylesheet" href="/css/bootstrap.min.css" />
<link rel="stylesheet" href="https://cdn.datatables.net/v/dt/dt-1.10.16/datatables.min.css" />
<script src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
<script src="//cdn.datatables.net/1.10.7/js/jquery.dataTables.min.js"></script>
<script type="text/javascript" src="/js/bootstrap.min.js"></script>
<script type="text/javascript">
    $(document).ready(function() {
        $('#table').DataTable();
    });
</script>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>

<body>
    <div th:replace="fragments/fragment :: header"></div>
    <h1 class="page-header">Update Penduduk</h1>

    <form action="/penduduk/update/submit" method="post" >
        <input type="hidden" name="nik" th:value="${penduduk.nik}" />
        <div>
            <label for="nama">Nama</label> <input type="text" name="nama" th:value="${penduduk.nama}" th:field="*{penduduk.nama}" />
        </div>
        <div>
            <label for="tempat_lahir">Tempat Lahir</label> <input type="text" name="tempat_lahir" th:value="${penduduk.tempat_lahir}" th:f
        </div>
        <div>
            <label for="tanggal_lahir">Tanggal Lahir</label> <input type="text" name="tanggal_lahir" th:value="${penduduk.tanggal_lahir}"
        </div>
        <div>
            <label for="golongan_darah">Golongan Darah</label>
                <select name = "golongan_darah" th:field="*{penduduk.golongan_darah}">
```

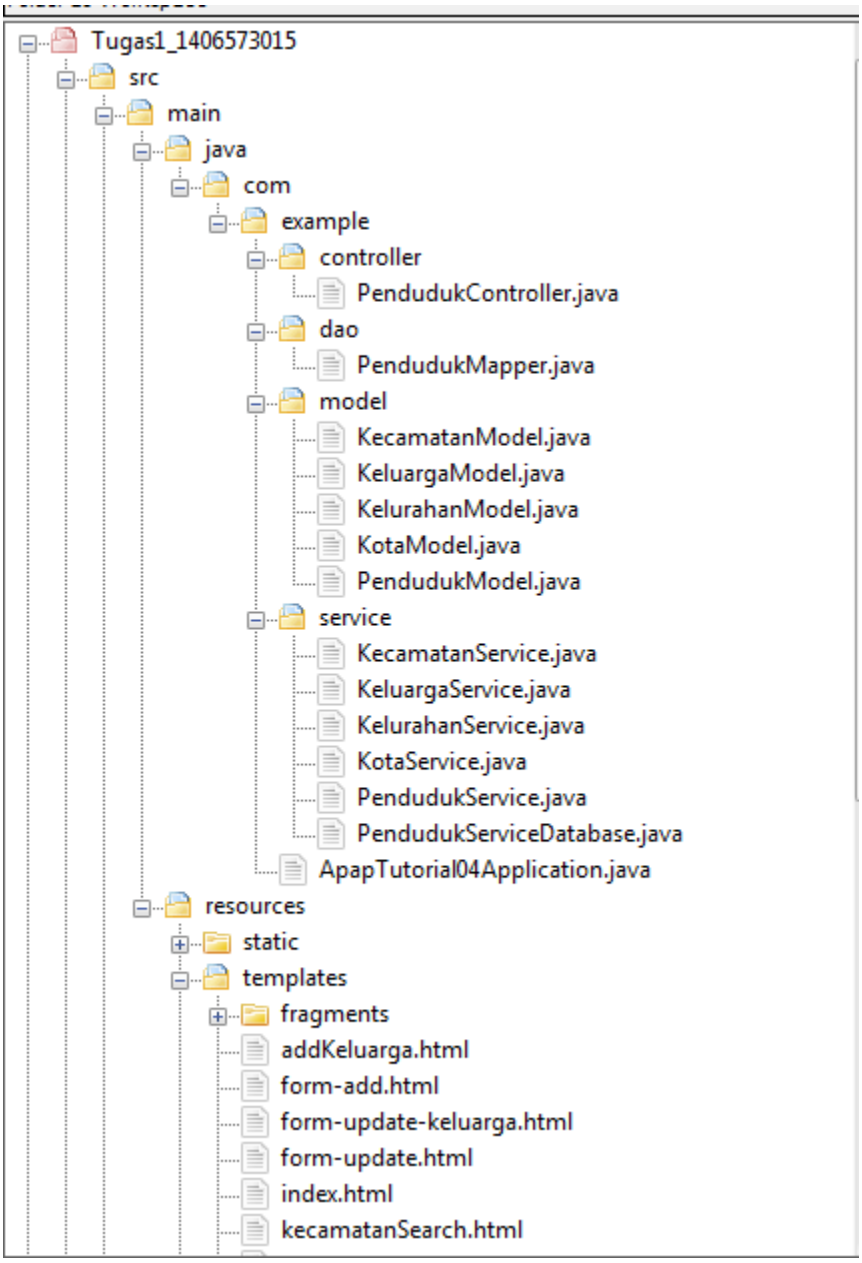
Mapping /submit digunakan untuk fungsi update ke database dan menampilkan halaman sukses

6. Update status kematian
    a. Menambahkan tombol ubah status pada penduduk?nik=
    b. Menambahkan fungsi di mapper
    c. Menambahkan fungsi servicedatabase
    d. Menambahkan fungsi di service
    e. Membuat view html
    f. Menambahkan membuat mapping
7. Mencari penduduk
    a. Membuat halaman html(4 halaman)
    b. Membuat mapping
    c. Menambahkan fungsi di mapper
    d. Menambahkan fungsi servicedatabase
    e. Menambahkan fungsi di service
    f. Implementasi datatable

Tampilan cari penduduk



Struktur File

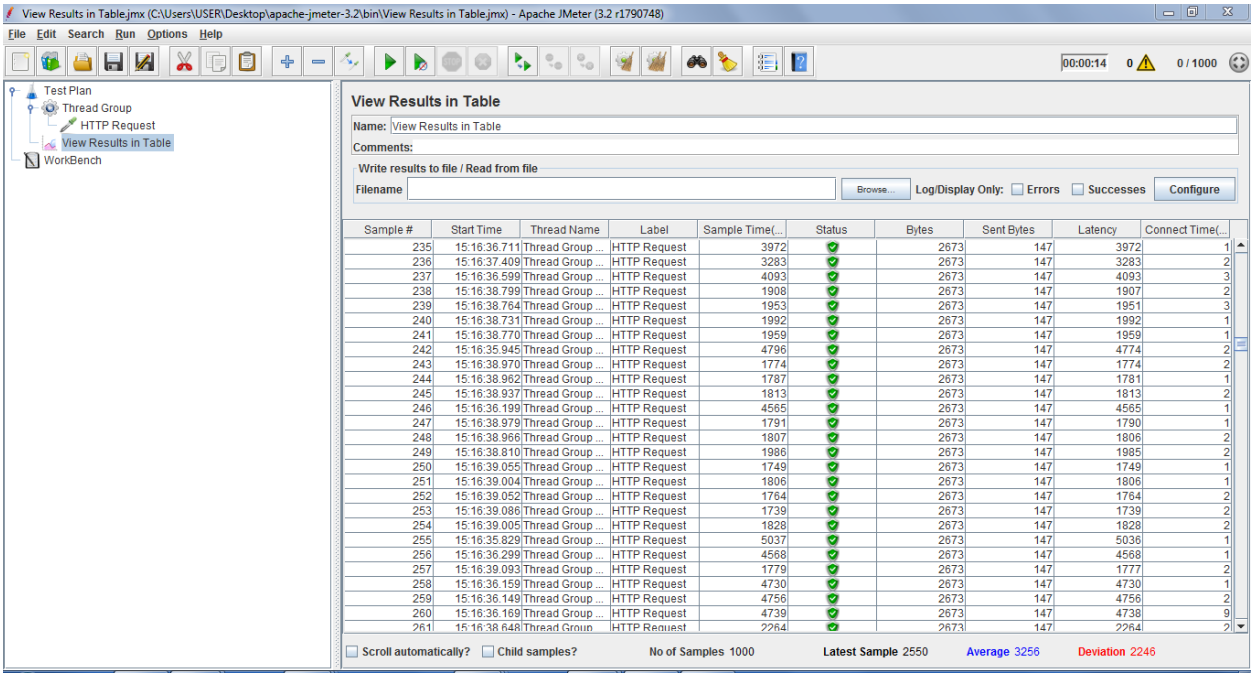


Dalam folder terdapat model, model merupakan sebuah objek yang merepresentasikan dan menyimpan informasi tertentu. Pada proyek ini ada 5 model yang dibuat sesuai dengan jumlah database dan setiap model mempunyai informasi yang berbeda. Kemudian terdapat service, service digunakan sebagai penghubung antara controller dan database. Didalam service terdapat fungsi logic untuk menipulasi suatu informasi tertentu dari view , shingga informasi tersebut bisa diolah oleh database, pada tugas yang dikerjakan hampir semua fungsi service-service yang penting ada dalam penduduk Servicedatabase. Sedangkan semua http request diolah pada penduduk controller. Controller berfungsi sebagai penghubung antara layer view dan service.

Kemudian terdapat mapper, mapper yang digunakan hanya ada 1 yaitu penduduk mapper. Mapper berfungsi untuk Create, Insert, Delete, Update. Kemudian ada view, semua view diletakan pada folder template. View di gunakan sebagai tampilan yang berfungsi sebagai media interaksi antara user dan sistem.

Stress Testing

Stress Testing dilakukan pada halaman http://localhost:8080/penduduk?nik=3101011112160001 menggunakan Jmeter. Berikut adalah hasil yang didapatkan



Dari hasil tersebut dapat dikatakan bahwa rata2 waktu yang diperlukan oleh user ketika mengkases halaman http://localhost:8080/penduduk?nik=3101011112160001 rata-rata 3256ms atau sekitar 3 detik ketika ada 1000 user yang mengakses bersamaan. Dari hasil ini dapat disimpulkan bahwa akses yang dilakukan tergolong cepat, karena hanya membutuhkan 3 detik ketika terdapat 1000 user yang mengakses secara bersamaan. Untuk melakukan optimasi yang dilakukan adalah membuat index pada tabel penduduk kolom id dan nik serta tabel keluarga kolom id.