

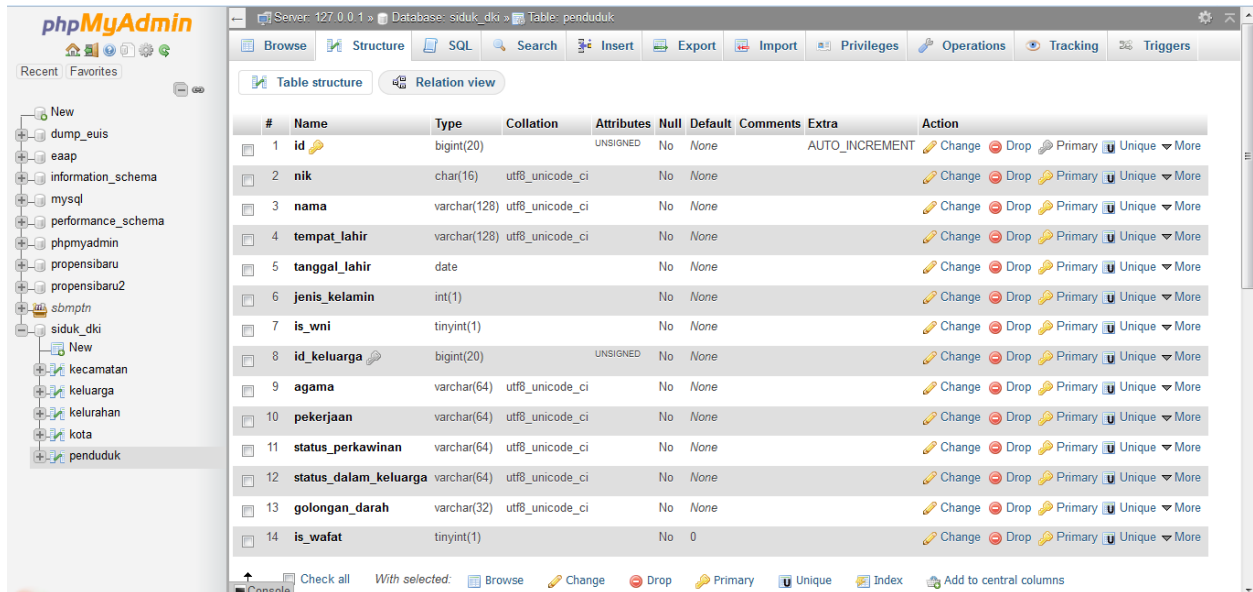
Iman Alfathan Yudhanto

1406623524

APAP-A

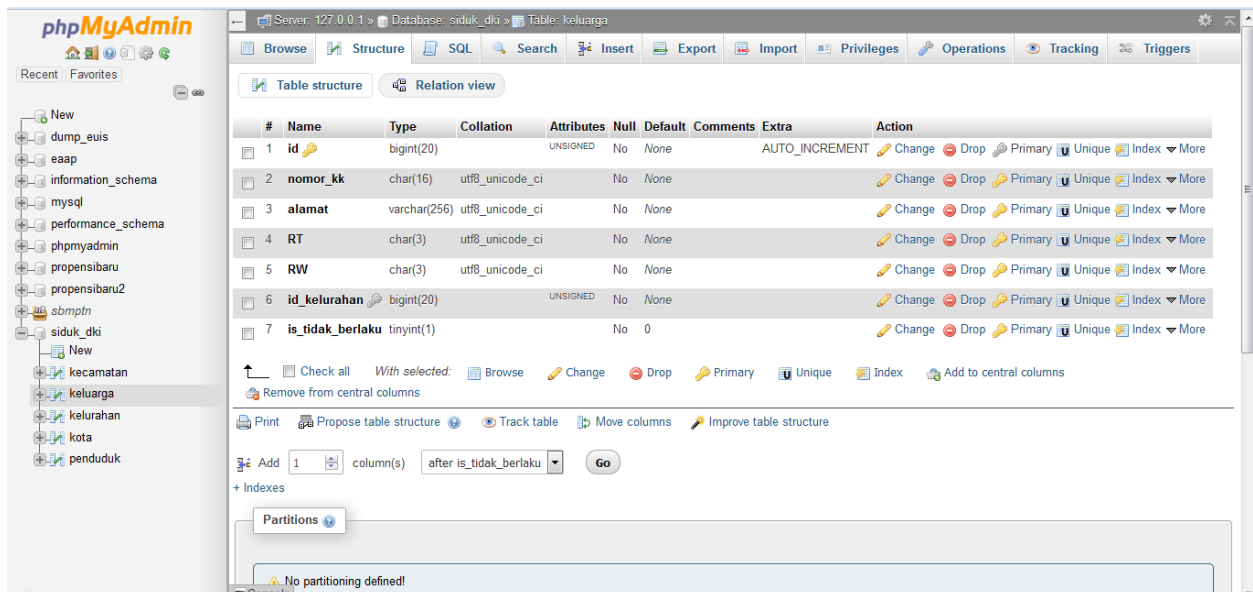
Tugas 1

Hal yang pertama dilakukan dalam mengembangkan tugas 1 adalah memperbaiki database. Database yang penulis miliki mendapatkan beberapa perubahan seperti membuat primary key di beberapa table dan membuat primary key-nya menjadi auto-increment.



The screenshot shows the phpMyAdmin interface with the 'penduduk' table selected. The table structure is displayed in 'Table structure' view. The table has 14 columns: id, nik, nama, tempat_lahir, tanggal_lahir, jenis_kelamin, is_wni, id_keluarga, agama, pekerjaan, status_perkawinan, status_dalam_keluarga, golongan_darah, and is_wafat. The 'id' column is the primary key and is set to AUTO_INCREMENT. The 'id_keluarga' column is also a primary key. The 'is_wafat' column has a default value of 0.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	nik	char(16)	utf8_unicode_ci		No	None			Change Drop Primary Unique More
3	nama	varchar(128)	utf8_unicode_ci		No	None			Change Drop Primary Unique More
4	tempat_lahir	varchar(128)	utf8_unicode_ci		No	None			Change Drop Primary Unique More
5	tanggal_lahir	date			No	None			Change Drop Primary Unique More
6	jenis_kelamin	int(1)			No	None			Change Drop Primary Unique More
7	is_wni	tinyint(1)			No	None			Change Drop Primary Unique More
8	id_keluarga	bigint(20)		UNSIGNED	No	None			Change Drop Primary Unique More
9	agama	varchar(64)	utf8_unicode_ci		No	None			Change Drop Primary Unique More
10	pekerjaan	varchar(64)	utf8_unicode_ci		No	None			Change Drop Primary Unique More
11	status_perkawinan	varchar(64)	utf8_unicode_ci		No	None			Change Drop Primary Unique More
12	status_dalam_keluarga	varchar(64)	utf8_unicode_ci		No	None			Change Drop Primary Unique More
13	golongan_darah	varchar(32)	utf8_unicode_ci		No	None			Change Drop Primary Unique More
14	is_wafat	tinyint(1)			No	0			Change Drop Primary Unique More



The screenshot shows the phpMyAdmin interface with the 'keluarga' table selected. The table structure is displayed in 'Table structure' view. The table has 7 columns: id, nomor_kk, alamat, RT, RW, id_kelurahan, and is_tidak_berlaku. The 'id' column is the primary key and is set to AUTO_INCREMENT. The 'id_kelurahan' column is also a primary key. The 'is_tidak_berlaku' column has a default value of 0.

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop Primary Unique Index More
2	nomor_kk	char(16)	utf8_unicode_ci		No	None			Change Drop Primary Unique Index More
3	alamat	varchar(256)	utf8_unicode_ci		No	None			Change Drop Primary Unique Index More
4	RT	char(3)	utf8_unicode_ci		No	None			Change Drop Primary Unique Index More
5	RW	char(3)	utf8_unicode_ci		No	None			Change Drop Primary Unique Index More
6	id_kelurahan	bigint(20)		UNSIGNED	No	None			Change Drop Primary Unique Index More
7	is_tidak_berlaku	tinyint(1)			No	0			Change Drop Primary Unique Index More

Setelah memperbaiki dan mengkoneksi database, maka penulis langsung berurusan dengan program. Yang pertama kali dibuat adalah membuat model berdasarkan database.

```

1 package com.example.demo.model;
2
3+ import lombok.AllArgsConstructor;
4
5
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class PendudukModel {
11     private Integer id;
12     private Integer idKeluarga;
13     private String nik;
14     private String nama;
15     private String tempatLahir;
16     private String tanggalLahir;
17     private Integer jenisKelamin;
18     private Integer isWNI;
19     private String agama;
20     private String pekerjaan;
21     private String statusPerkawinan;
22     private String statusDalamKeluarga;
23     private String golonganDarah;
24     private Integer isWafat;
25     private KeluargaModel keluarga;
26 }

```

```

1 package com.example.demo.model;
2
3+ import lombok.AllArgsConstructor;
4
5
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class KelurahanModel {
11     private Integer id;
12     private Integer idKecamatan;
13     private String kodeKelurahan;
14     private String namaKelurahan;
15     private String kodePos;
16     private KecamatanModel kecamatan;
17 }
18

```

```

1 package com.example.demo.model;
2
3+ import lombok.AllArgsConstructor;
4
5
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class KecamatanModel {
11     private Integer id;
12     private Integer idKota;
13     private String kodeKecamatan;
14     private String namaKecamatan;
15     private KotaModel kota;
16 }
17

```

```

1 package com.example.demo.model;
2
3+ import lombok.AllArgsConstructor;
4
5
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class KotaModel {
11     private Integer id;
12     private String kodeKota;
13     private String namaKota;
14 }
15

```

```

1 package com.example.demo.model;
2
3+ import lombok.AllArgsConstructor;
4
5
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class KeluargaModel {
11     private Integer id;
12     private Integer idKelurahan;
13     private String nomorKK;
14     private String alamat;
15     private String rt;
16     private String rw;
17     private Integer isTidakBerlaku;
18     private KelurahanModel kelurahan;
19 }
20

```

Setelah membuat fitur, penulis memulai membuat fitur-fitur yang disuruh pada soal.

[1] Tampilkan Data Penduduk Berdasarkan NIK

Initial Page: /

Fitur ini berfungsi dalam menampilkan penduduk yang dicari sesuai dengan nik-nya. Yang pertama penulis lakukan adalah membuat query untuk mencari data penduduk berdasarkan nik pada pendudukMapper. Penulis melakukan pemanggilan mapper beruntun dari table kota hingga table keluarga. Tiap table diberi result property yang memanggil fungsi mapper sebelumnya.

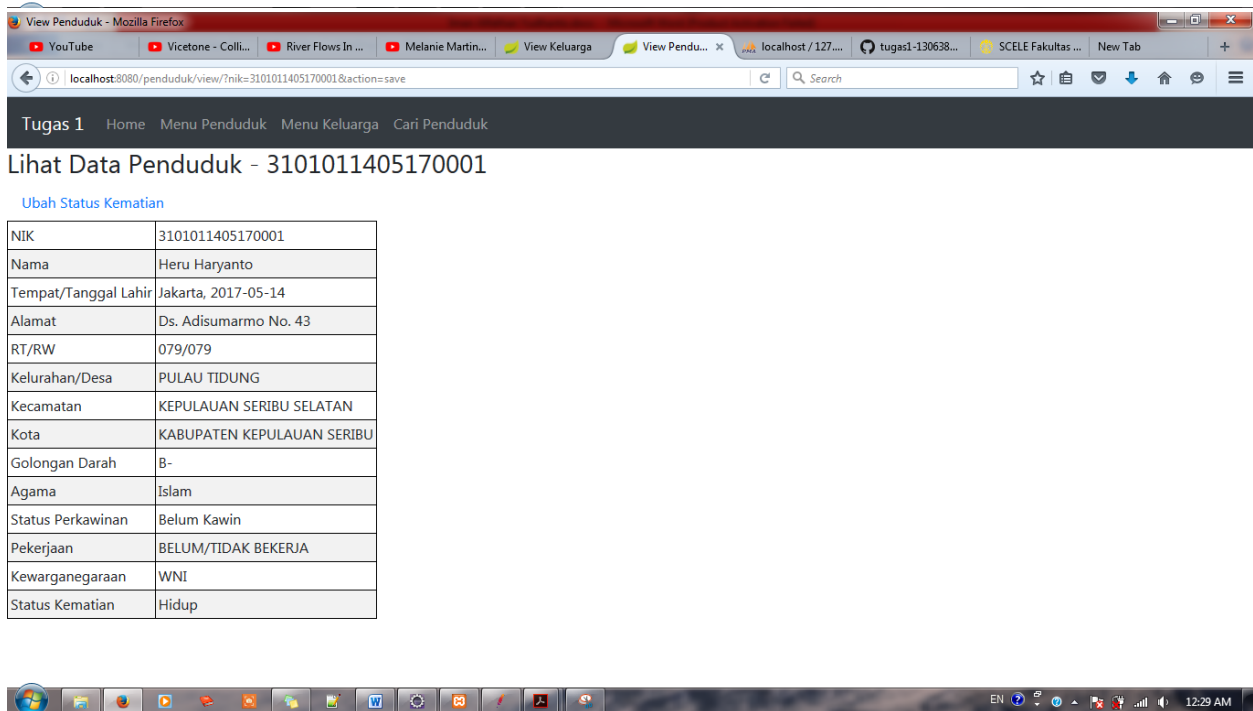
```
44
23 @Select("Select * from kota " + "where id=#{id}")
24 @Results(value = { @Result(property = "namaKota", column = "nama_kota"),
25                  @Result(property = "kodeKota", column = "kode_kota") })
26 KotaModel selectKota(@Param("id") String id);
27
28 @Select("SELECT distinct kecamatan.id, kecamatan.id_kota as id_kota, kecamatan.nama_kecamatan, "
29         + "kecamatan.kode_kecamatan as kode_kecamatan "
30         + "from kecamatan " + "join kota on kecamatan.id_kota=kota.id " + "where kecamatan.id=#{idKecamatan}")
31 @Results(value = { @Result(property = "idKecamatan", column = "id_kecamatan"),
32                  @Result(property = "namaKecamatan", column = "nama_kecamatan"),
33                  @Result(property = "kodeKecamatan", column = "kode_kecamatan"),
34                  @Result(property = "kota", column = "id_kota", javaType = KotaModel.class, many = @Many(select = "selectKota")) })
35 KecamatanModel selectKecamatan(@Param("idKecamatan") String idKecamatan);
36
37 @Select("SELECT distinct kelurahan.id, kelurahan.id_kecamatan as id_kecamatan, kelurahan.nama_kelurahan as nama_kelurahan, "
38         + "kelurahan.kode_kelurahan as kode_kelurahan "
39         + "from kelurahan " + "join keluarga on kelurahan.id = keluarga.id_kelurahan "
40         + "where kelurahan.id=#{idKelurahan}")
41 @Results(value = { @Result(property = "idKecamatan", column = "id_kecamatan"),
42                  @Result(property = "namaKelurahan", column = "nama_kelurahan"),
43                  @Result(property = "kodeKelurahan", column = "kode_kelurahan"),
44                  @Result(property = "kecamatan", column = "id_kecamatan", javaType = KecamatanModel.class, many = @Many(select = "selectKecamatan")) })
45 KelurahanModel selectKelurahan(@Param("idKelurahan") String idKelurahan);
46
47 @Select("SELECT distinct keluarga.id, keluarga.id_kelurahan as id_kelurahan, keluarga.alamat as alamat, keluarga.rt as rt, "
48         + "keluarga.rw as rw, keluarga.nomor_kk as nomor_kk "
49         + "from keluarga " + "join penduduk on keluarga.id = penduduk.id_keluarga "
50         + "where keluarga.id=#{idKeluarga}")
51 @Results(value = { @Result(property = "idKelurahan", column = "id_kelurahan"),
52                  @Result(property = "alamat", column = "alamat"), @Result(property = "rt", column = "rt"),
53                  @Result(property = "rw", column = "rw"), @Result(property = "nomorKK", column = "nomor_kk"),
54                  @Result(property = "kelurahan", column = "id_kelurahan", javaType = KelurahanModel.class, many = @Many(select = "selectKelurahan")) })
55 KeluargaModel selectKeluarga(@Param("idKeluarga") String idKeluarga);
56
57 @Select("select * from penduduk where nik = #{nik}")
58 @Results(value = { @Result(property = "tempatLahir", column = "tempat_lahir"),
59                  @Result(property = "tanggalLahir", column = "tanggal_lahir"),
60                  @Result(property = "golonganDarah", column = "golongan_darah"),
61                  @Result(property = "jenisKelamin", column = "jenis_kelamin"),
62                  @Result(property = "idKeluarga", column = "id_keluarga"),
63                  @Result(property = "statusDalamKeluarga", column = "status_dalam_keluarga"),
64                  @Result(property = "statusPerkawinan", column = "status_perkawinan"),
65                  @Result(property = "isWNI", column = "is_wni"), @Result(property = "isWafat", column = "is_wafat"),
66                  @Result(property = "keluarga", column = "id_keluarga", javaType = KeluargaModel.class, many = @Many(select = "selectKeluarga")) })
67 PendudukModel selectPenduduk(@Param("nik") String nik);
```

Penulis melakukan pemanggilan mapper beruntun dari table kota hingga table keluarga. Tiap table diberi result property yang memanggil fungsi mapper sebelumnya. Lalu penulis membuat fungsi select penduduk yang berguna untuk mengambil penduduk berdasarkan nik-nya. Jadi penulis menggunakan query beruntun untuk memanggil sebuah query agar bisa di-reuse. Setelah membuat query, penulis menggunakan controller, service, dan controller untuk menampilkan dan menghubungkan dengan front-end. Setelah itu, penulis dapat menampilkan tampilan lihat penduduk by nik.

Tambah Penduduk

Cari Penduduk

Ubah Penduduk



[2] Tampilkan Data Keluarga Beserta Daftar Anggotanya Berdasarkan Nomor KK

Fitur ini berfungsi dalam menampilkan keluarga yang dicari sesuai dengan nkk-nya. Yang pertama penulis lakukan adalah membuat query untuk mencari data keluarga berdasarkan nik pada pendudukMapper. Penulis melakukan pemanggilan mapper beruntun dari table kota hingga table keluarga. Tiap table diberi result property yang memanggil fungsi mapper sebelumnya.

```

19 @Mapper
20 public interface KeluargaMapper {
21     @Select("Select * from kota "
22             + "where id=#{id}")
23     @Results(value = {
24         @Result (property = "namaKota", column = "nama_kota")
25     })
26     KotaModel selectKota(@Param("id") String id);
27
28     @Select("SELECT distinct kecamatan.id, kecamatan.id_kota as id_kota, kecamatan.nama_kecamatan as nama_kecamatan, "
29             + "kecamatan.kode_kecamatan as kode_kecamatan "
30             + "from kecamatan "
31             + "join kota on kecamatan.id_kota=kota.id "
32             + "where kecamatan.id=#{idKecamatan}")
33     @Results(value = {
34         @Result (property = "idKecamatan", column = "id_kecamatan"),
35         @Result (property = "namaKecamatan", column = "nama_kecamatan"),
36         @Result (property = "kodeKecamatan", column = "kode_kecamatan"),
37         @Result (property = "kota", column = "id_kota", javaType = KotaModel.class, many = @Many(select="selectKota"))
38     })
39     KecamatanModel selectKecamatan(@Param("idKecamatan") String idKecamatan);
40
41     @Select("SELECT distinct kelurahan.id, kelurahan.id_kecamatan as id_kecamatan, kelurahan.nama_kelurahan as nama_kelurahan "
42             + "from kelurahan "
43             + "join keluarga on kelurahan.id = keluarga.id_kelurahan "
44             + "where kelurahan.id=#{idKelurahan}")
45     @Results(value = {
46         @Result (property = "idKecamatan", column = "id_kecamatan"),
47         @Result (property = "namaKelurahan", column = "nama_kelurahan"),
48         @Result (property = "kecamatan", column = "id_kecamatan", javaType = KecamatanModel.class, many = @Many(select="selectKecamatan"))
49     })
50     KelurahanModel selectKelurahan(@Param("idKelurahan") String idKelurahan);
51
52     @Select("SELECT distinct keluarga.id, keluarga.id_kelurahan as id_kelurahan, keluarga.alamat as alamat, "
53             + "keluarga.rt as rt, keluarga.rw as rw, keluarga.nomor_kk as nomor_kk "
54             + "from keluarga "
55             + "left join penduduk on keluarga.id = penduduk.id_keluarga "
56             + "where keluarga.nomor_kk=#{nkk}")
57     @Results(value = {
58         @Result (property = "idKelurahan", column = "id_kelurahan"),
59         @Result (property = "alamat", column = "alamat"),
60         @Result (property = "rt", column = "rt"),
61         @Result (property = "rw", column = "rw"),
62         @Result (property = "nomorKK", column = "nomor_kk"),
63         @Result (property = "kelurahan", column = "id_kelurahan", javaType = KelurahanModel.class, many = @Many(select="selectKelurahan"))
64     })
65     KeluargaModel selectKeluarga(@Param("nkk") String nkk);
66 }

```

Penulis melakukan pemanggilan mapper beruntun dari table kota hingga table keluarga. Penulis menggunakan query beruntun untuk memanggil sebuah query agar bisa di-reuse. Tiap table diberi result property yang memanggil fungsi mapper sebelumnya. Gunanya adalah untuk mengambil data keluarga sesuai yang disuruh di soal.

```

67     @Select("SELECT p.nama, p.nik as nik, p.jenis_kelamin as jenis_kelamin, p.tempat_lahir as tempat_lahir, p.tanggal_lahir as tanggal_lahir, "
68             + "p.agama as agama, p.pekerjaan as pekerjaan, p.status_perkawinan as status_perkawinan, "
69             + "p.status_dalam_keluarga as status_dalam_keluarga, p.is_wni as is_wni, p.is_wafat as is_wafat "
70             + "FROM penduduk p, keluarga k "
71             + "WHERE k.id=p.id_keluarga and k.nomor_kk =#{nkk}")
72     @Results(value = {
73         @Result (property = "jenisKelamin", column = "jenis_kelamin"),
74         @Result (property = "tempatLahir", column = "tempat_lahir"),
75         @Result (property = "tanggalLahir", column = "tanggal_lahir"),
76         @Result (property = "statusPerkawinan", column = "status_perkawinan"),
77         @Result (property = "isWafat", column = "is_wafat"),
78         @Result (property = "statusDalamKeluarga", column = "status_dalam_keluarga"),
79         @Result (property = "isWNI", column = "is_wni")
80     })
81
82     ArrayList<PendudukModel> selectPendudukByKeluarga(@Param("nkk") String nkk);

```

Lalu penulis membuat fungsi selectPendudukByKeluarga yang berguna untuk mengambil penduduk berdasarkan id keluarganya. Setelah membuat query, penulis menggunakan controller, service, dan controller agar bisa ditampilkan dan terhubung dengan front-end. Setelah itu, penulis dapat menampilkan tampilan viewKeluargaByNKK seperti gambar di bawah.

Tambah Keluarga

Cari Keluarga

Ubah Keluarga

Lihat Data Keluarga - 3101010104070002

NKK:	3101010104070002
Alamat:	Jln. Sutami No. 781
RT/RW:	069/033
Kelurahan/Desa:	PULAU UNTUNG JAWA
Kecamatan:	KEPULAUAN SERIBU SELATAN
Kota:	KABUPATEN KEPULAUAN SERIBU

Anggota Keluarga

No	Nama Lengkap	NIK	Jenis Kelamin	Tempat Lahir	Tanggal Lahir	Agama	Pekerjaan	Status Perkawinan	Status Dalam Keluarga	Kewarganegaraan
1	Dina Meja Calik	3101019002020001	Wanita	Hogwart	1890-02-02	Islan	pembasmi kejahatan	Kawin	Istri Immortal	WNI
2	Dijeuna Tina Singkong	3101019002020002	Wanita	Jakarta	1890-02-02	Islan	pembasmi kejahatan	-	Saudara Kembar	WNI
3	Udin Ulin di Buruan	3101018902020002	Pria	Jakarta	1990-02-21	Islam KTP	Juru Masak	Sudah Menikah	Suami muda 2	WNI
4	Abdul Bedul Bedebuk	3101018902020001	Pria	Jakarta	1989-02-02	Islam	pembasmi kejahatan	Sudah Menikah	Suami muda	WNI
5	Rampes Sampurasun	3101010009180001	Pria	Bandung	2010-09-18	Muslim	pelajar	Kawin	Anak	WNI
6	Sirah Lengeunna Bodas	3101019607070001	Wanita	Citayam	1996-07-07	Indomie	Mahasiswa	Belum Kawin	Adik Perempuan	WNI

[3] Menambahkan Penduduk Baru Sebagai Anggota Keluarga

Fitur ini berfungsi dalam menambahkan data penduduk. Yang pertama penulis lakukan adalah membuat query untuk menambahkan penduduk berdasarkan kriteria di soal.

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

Query tersebut berfungsi untuk menambahkan data sesuai kriteria. Pada bagian values, nilai nik diisi dengan nilai "tes". Tujuannya adalah memberi nilai nik temporal, setelah itu ada pemrosesan pada controller untuk mengupdate nik-nya sesuai dengan permintaan di soal.

```

@Select("SELECT * FROM penduduk ORDER BY id DESC LIMIT 1")
@Results(value = { @Result(property = "tempatLahir", column = "tempat_lahir"),
    @Result(property = "tanggalLahir", column = "tanggal_lahir"),
    @Result(property = "golonganDarah", column = "golongan_darah"),
    @Result(property = "jenisKelamin", column = "jenis_kelamin"),
    @Result(property = "idKeluarga", column = "id_keluarga"),
    @Result(property = "statusDalamKeluarga", column = "status_dalam_keluarga"),
    @Result(property = "statusPerkawinan", column = "status_perkawinan"),
    @Result(property = "isWNI", column = "is_wni"), @Result(property = "isWafat", column = "is_wafat") })
PendudukModel lastPenduduk();

@Update("UPDATE penduduk SET nik = #{nik} where penduduk.id=#{id}")
void updateNIK(PendudukModel penduduk);

```

Berikut adalah query yang digunakan untuk mengubah nik penduduk yang telah ditambahkan. Method lastPenduduk berfungsi untuk mendapatkan penduduk terakhir yang telah dimasukkan. Sedangkan method updateNIK berfungsi untuk mengganti NIK temporal penduduk dengan NIK baru sesuai kriteria soal. Berikut adalah tampilan ketika menambahkan penduduk.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Tambah Penduduk

Cari Penduduk

Ubah Penduduk

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Tambah Penduduk

Masukkan Nama

Tempat Lahir

Tanggal Lahir

Golongan Darah

Jenis Kelamin

Agama

Pendidikan

Kewarganegaraan

Status Kematian

Status Perkawinan

Status Dalam Keluarga

ID Keluarga

Drag the cursor around the area you want to capture.

Sukses!

Penduduk dengan nik 3101019604180001 Berhasil Ditambahkan

Gambar di atas adalah tampilan ketika berhasil menambahkan penduduk.

[4] Menambahkan Keluarga Baru

Fitur ini berfungsi dalam menambahkan data keluarga. Yang pertama penulis lakukan adalah membuat query untuk menambahkan keluarga berdasarkan kriteria di soal.

```
@Insert("INSERT INTO keluarga ( nomor_kk, alamat, RT, RW, id_kelurahan, is_tidak_berlaku) "
        + "VALUES ({nomorKK}, #{alamat}, #{rt}, #{rw}, #{idKelurahan}, 0)")
void addKeluarga(KeluargaModel keluarga);
```

Query tersebut berfungsi untuk menambahkan data sesuai kriteria. Kolom yang akan diinsert secara langsung di webnya adalah alamat, rt, rw, dan idKelurahan. Nilai nomorKK secara otomatis diisi nilai 0 (yang menandakan keluarga tersebut aktif). Untuk nomorKK, akan diisi nilai sesuai dengan proses pemberian nilai nomorKK pada controller.

```
27 @RequestMapping("/keluarga")
28 public String viewHalamanKeluargak(Model model) {
29     return "index-keluarga";
30 }
31
32 @RequestMapping("/keluarga/tambah")
33 public String add(Model model) {
34     KeluargaModel keluarga = new KeluargaModel();
35     model.addAttribute("keluarga", keluarga);
36     return "form-add-keluarga";
37 }
```



```

39 @PostMapping("/keluarga/tambah/submit")
40 public String addSubmit(Model model, @ModelAttribute KeluargaModel keluargaModel) {
41     //
42     // Ngambil variabel terakhir buat compare tanggal
43     System.out.println("udah diinsert " + keluargaModel);
44     KeluargaModel keluargaTerakhir = keluargaService.keluargaTerakhir();
45     String tanggalAkhir = keluargaTerakhir.getNomorKK().substring(6, 12);
46     System.out.println("keluarga terakhir " + keluargaTerakhir + tanggalAkhir);
47
48     // bikin tanggal
49     DateTimeFormatter dtf = DateTimeFormatter.ofPattern("dd/MM/yyyy");
50     LocalDate localDate = LocalDate.now();
51     String tanggal = dtf.format(localDate);
52     System.out.println("tanggal " + tanggal);
53     //
54     String nkkAsli = "";
55     // compare tanggal untuk buat nkk
56     if (tanggalAkhir.equalsIgnoreCase(tanggal)) {
57         Integer nkkAkhir = Integer.parseInt(keluargaTerakhir.getNomorKK().substring(12, 16));
58         nkkAkhir = nkkAkhir + 1;
59         String count = new DecimalFormat("0000").format(nkkAkhir);
60         nkkAsli = keluargaTerakhir.getKelurahan().getKecamatan().getKodeKecamatan().substring(0, 6) + tanggal
61             + count;
62     } else {
63         nkkAsli = keluargaTerakhir.getKelurahan().getKecamatan().getKodeKecamatan().substring(0, 6) + tanggal
64             + "0001";
65     }
66     // String nkk = camat.substring(0, camat.length()-1);
67     System.out.println("nkk " + nkkAsli);
68     keluargaModel.setNomorKK(nkkAsli);
69     keluargaService.addKeluarga(keluargaModel);
70     model.addAttribute("nomorKK", nkkAsli);
71     return "success-add-keluarga";
72 }

```

Pada gambar di atas, terdapat cara memberi nilai nomorKK berdasarkan kriteria pada soal.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Tambah Keluarga

Alamat

RT

RW

ID Kelurahan

Pada gambar di atas adalah gambar form menambahkan keluarga. Yang ditambahkan adalah alamat, rt, rw, dan idKelurahan. Alasan kenapa penambahan kota dan kecamatan tidak dilakukan adalah karena ketika memasukkan idKelurahan, kita sudah mendapatkan Kecamatan dan kota secara langsung dan menghindari pemasukkan idKecamatan dan idKota yang asal.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Sukses!

Penduduk dengan NKK 3171052210170001 Berhasil Ditambahkan

Gambar di atas adalah proses dimana keluarga baru berhasil ditambahkan.

[5] Mengubah Data Penduduk

Fitur ini berfungsi untuk mengubah data penduduk yang ada. Yang pertama penulis lakukan adalah membuat query untuk mengubah penduduk berdasarkan kriteria di soal.

```
78 @Update("UPDATE penduduk "  
79 + "SET nama=#{nama}, tempat_lahir=#{tempatLahir}, tanggal_lahir=#{tanggalLahir}, "  
80 + "golongan_darah=#{golonganDarah}, agama=#{agama}, jenis_kelamin=#{jenisKelamin}, pekerjaan=#{pekerjaan}, "  
81 + "is_wni=#{isWNI}, status_dalam_keluarga=#{statusDalamKeluarga}, status_perkawinan=#{statusPerkawinan}, id_keluarga=#{idKeluarga} "  
82 + "WHERE nik = #{nik}")  
83 void updatePenduduk(PendudukModel penduduk);  
84  
85 @Update("UPDATE penduduk " + "SET is_wafat= #{isWafat} " + "WHERE nik = #{nik}")  
86 void updateKematian(PendudukModel penduduk);
```

Query tersebut berfungsi untuk mengubah data sesuai kriteria.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Update Penduduk

Nama
Elmo

NIK
3101028211180001

Tempat Lahir
Meikarta

Tanggal Lahir
1982-11-18

Golongan Darah
O+

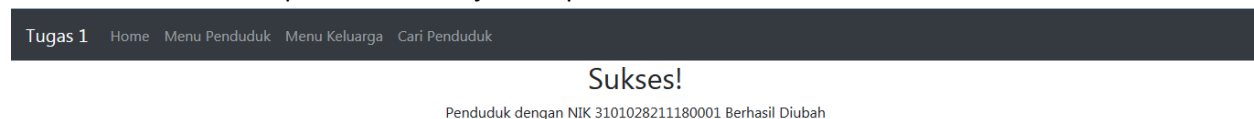
Jenis Kelamin
Pria

Gambar di atas adalah gambar form ketika ingin mengubah data penduduk. Nilai nik pada form tidak dapat diubah.

Data not found

NIK = 3101

Gambar di atas adalah pemberitahuan jika nik penduduk tidak ditemukan.



Gambar di atas adalah gambar ketika berhasil mengubah data penduduk.

[6] Mengubah Data Keluarga

Fitur ini berfungsi untuk mengubah data keluarga. Yang pertama penulis lakukan adalah membuat query untuk mengubah keluarga berdasarkan kriteria di soal.

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

Query tersebut berfungsi untuk mengubah data sesuai kriteria.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

NKK

3171052010170001

Alamat

Jalan Sesama Trans 7

RT

1

RW

1

ID Kelurahan

4

Save

Gambar di atas adalah gambar ketika ingin mengubah data penduduk. Hanya kolom NKK yang nilainya tidak bisa diubah.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Sukses!

Keluarga dengan NKK 3171052010170001 Berhasil Diubah

Gambar di atas adalah status ketika data keluarga berhasil dirubah.

[7] Mengubah Status Kematian Penduduk

Fitur ini berguna untuk mengubah status kematian seorang penduduk. Yang pertama penulis lakukan adalah membuat query untuk mengubah status kematian seorang penduduk berdasarkan kriteria di soal.

```
@Update("UPDATE penduduk " + "SET is_wafat= #{isWafat} " + "WHERE nik = #{nik}")
void updateKematian(PendudukModel penduduk);
```

Query sederhana tersebut berguna untuk mengganti status kematian seorang penduduk berdasarkan nik. Selain itu, penulis menambahkan program pada controller untuk menentukan isTidakBerlaku pada data Keluarga.

```

153 @PostMapping(value = "/penduduk/submit")
154 public String updateKematian(Model model, @ModelAttribute PendudukModel penduduk) {
155     //
156     pendudukService.updateKematian(penduduk);
157     System.out.println("penduduk edit" + penduduk);
158     PendudukModel pendudukLama = pendudukService.selectPendudukbyNIK(penduduk.getNik());
159     System.out.println("pendudukLama " + pendudukLama);
160     ArrayList<PendudukModel> anggotaKeluarga = keluargaService
161         .selectPendudukbyKeluarga(pendudukLama.getKeluarga().getNomorKK());
162     System.out.println("anggota keluarga " + anggotaKeluarga);
163     int counterKematian = 0;
164     for (int i = 0; i < anggotaKeluarga.size(); i++) {
165         if (anggotaKeluarga.get(i).getIsWafat() == 1) {
166             counterKematian++;
167             System.out.println(counterKematian);
168             System.out.println("size " + anggotaKeluarga.size());
169         }
170     }
171     if (counterKematian == anggotaKeluarga.size()) {
172         System.out.println("semua anggota keluarga mati. NKK tidak berlaku lagi." + counterKematian + " "
173             + anggotaKeluarga.size());
174         pendudukLama.getKeluarga().setIsTidakBerlaku(1);
175         keluargaService.updateNKK(pendudukLama.getKeluarga());
176     }
177     //
178     model.addAttribute("penduduk", penduduk);
179     return "success-update-kematian";
180 }

```

Pada bagian controller, terdapat sebuah arraylist yang berasal dari hasil method keluarga service. Lalu ada integer countkematian untuk menghitung/sebagai tanda berapa banyak keluarga yang meninggal. Setelah itu dilakukan looping untuk pengecekan. Jika counter jumlahKematian sama dengan anggota keluarga yang sbelumnya telah didapatkan, maka dilakukan pengupdate-an isStatusTidakBerlaku.

```

111 @Update("UPDATE keluarga "
112     + "SET is_tidak_berlaku=#{isTidakBerlaku} "
113     + "WHERE nomor_kk = #{nomorKK}")
114 void updateNKK(KeluargaModel keluarga);
115 }

```

Gambar di atas adalah query yang berfungsi untuk meng-update isTidakBerlaku berdasarkan NKK yang dicari.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Update Status Wafat Penduduk

Nama

Elmo

NIK

3101028211180001

Status Kematian

Hidup

Update

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Sukses!

Penduduk dengan NIK 3101028211180001 Sudah Tidak Aktif

Gambar di atas adalah status ketika status kematian penduduk berhasil dirubah.

[8] Tampilkan Data Penduduk Berdasarkan Kota/Kabupaten, Kecamatan, dan Kelurahan Tertentu

Fitur ini berfungsi dalam menampilkan penduduk yang dicari sesuai dengan kota, kecamatan, dan kelurahan dimana penduduk itu tinggal. Yang pertama penulis lakukan adalah membuat query untuk mencari seluruh kota, kecamatan, dan kelurahan untuk dimasukkan ke multivalue pada front end.

```
99 @Select("Select * from kota order by nama_kota")
100 @Results(value = { @Result(property = "id", column = "id"), @Result(property = "namaKota", column = "nama_kota"),
101 @Result(property = "kodeKota", column = "kode_kota") })
102 ArrayList<KotaModel> selectListKota();
103
104 @Select("SELECT distinct kecamatan.id, kecamatan.id_kota as id_kota, kecamatan.nama_kecamatan as nama_kecamatan, "
105 + "kecamatan.kode_kecamatan as kode_kecamatan "
106 + "from kecamatan " + "join kota on kecamatan.id_kota=kota.id order by nama_kecamatan")
107 @Results(value = { @Result(property = "id", column = "id"),
108 @Result(property = "idKecamatan", column = "id_kecamatan"),
109 @Result(property = "namaKecamatan", column = "nama_kecamatan"),
110 @Result(property = "kodeKecamatan", column = "kode_kecamatan") })
111 ArrayList<KecamatanModel> selectListKecamatan(@Param("idKota") Integer idKota);
112
113 @Select("SELECT distinct kelurahan.id, kelurahan.id_kecamatan as id_kecamatan, kelurahan.nama_kelurahan as nama_kelurahan, "
114 + "kelurahan.kode_kelurahan as kode_kelurahan "
115 + "from kelurahan " + "join keluarga on kelurahan.id = keluarga.id_kelurahan order by nama_kelurahan ")
116 @Results(value = { @Result(property = "id", column = "id"),
117 @Result(property = "idKecamatan", column = "id_kecamatan"),
118 @Result(property = "namaKelurahan", column = "nama_kelurahan"),
119 @Result(property = "kodeKelurahan", column = "kode_kelurahan") })
120 ArrayList<KelurahanModel> selectListKelurahan(@Param("idKecamatan") Integer idKecamatan);
```

Query di atas berfungsi untuk mengambil data kota, kecamatan, dan kelurahan yang diurutkan secara alfabetikal.

```
122 @Select("select distinct penduduk.nik as nik, penduduk.nama as nama, penduduk.jenis_kelamin as kelamin, kota.nama_kota as nama_kota, "
123 + "kecamatan.nama_kecamatan as nama_kec, kelurahan.nama_kelurahan as nama_kel "
124 + "from penduduk, kecamatan, kelurahan, kota, keluarga " + "where penduduk.id_keluarga = keluarga.id "
125 + "and keluarga.id_kelurahan = kelurahan.id " + "and kelurahan.id_kecamatan = kecamatan.id "
126 + "and kecamatan.id_kota = kota.id "
127 + "and kota.id=#{idKota} and kelurahan.id=#{idKel} and kecamatan.id=#{idKecamatan}")
128 @Results(value = { @Result(property = "jenisKelamin", column = "kelamin"),
129 @Result(property = "keluarga.kelurahan.kecamatan.kota.namaKota", column = "nama_kota"),
130 @Result(property = "keluarga.kelurahan.kecamatan.namaKecamatan", column = "nama_kec"),
131 @Result(property = "keluarga.kelurahan.namaKelurahan", column = "nama_kel"),
132 })
133 ArrayList<PendudukModel> selectListPendudukbyCari(@Param("idKota") String idKota,
134 @Param("idKecamatan") String idKecamatan, @Param("idKel") String idKel);
```

Query di atas berfungsi untuk mengambil penduduk berdasarkan idkota, idkecamatan, dan idkelurahan.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk

Cari Penduduk

Kota

KABUPATEN KEPULAUAN SERIBU

Kecamatan

KEPULAUAN SERIBU SELATAN

Kelurahan

PULAU PARI

Save

Gambar di atas adalah tampilan multiselect pada fitur cariPenduduk.

Tugas 1 Home Menu Penduduk Menu Keluarga Cari Penduduk			
List Penduduk			
lihat penduduk di kota KABUPATEN KEPULAUAN SERIBU, Kecamatan KEPULAUAN SERIBU SELATAN, Kelurahan PULAU PARI			
No	NIK	Nama Lengkap	Jenis Kelamin
1	3101010303890001	Taswir Rajata	Pria
2	3101011408150001	Mulyono Hutagalung S.Gz	Pria
3	3101012309120001	Galar Permadi	Pria
4	3101011009870001	Himawan Garang Habibi M.Ak	Pria
5	3101010812740001	Argono Utama	Pria
6	3101010202170002	Gandewa Nyana Wibowo S.Ked	Pria
7	3101011411650001	Maman Drajat Saputra	Pria
8	3101010707910001	Cahyono Simbolon	Pria
9	3101010602830001	Makara Latupono	Pria

Gambar di atas adalah tampilan ketika berhasil mencari penduduk yang diinginkan.

ERROR!

Halaman yang dicari tidak ditemukan

Gambar di atas adalah tampilan ketika gagal mencari penduduk yang diinginkan.

[11] ****BONUS** Menambahkan Error Page**

Ketika user ingin mengakses halaman yang tidak ada atau melakukan kesalahan input, maka akan muncul pemberitahuan berupa halaman error.

ERROR!

Halaman yang dicari tidak ditemukan

Gambar tersebut merupakan halaman error yang muncul ketika url yang tidak dibuat pada controller diakses.

[12] ****BONUS** Fitur-Fitur Lain yang Mendukung Aplikasi**

Fitur tambahan yang ada dalam tugas 1 ini adalah:

- fitur search untuk melakukan pencarian dan perubahan penduduk.

Tugas 1 [Home](#) [Menu Penduduk](#) [Menu Keluarga](#) [Cari Penduduk](#)

Tambah Penduduk

Cari Penduduk

Ubah Penduduk

Fitur ini memudahkan ketika ingin mencari penduduk dan mengubah data penduduk, dibandingkan dengan mengetik langsung pada url.

- fitur search untuk melakukan pencarian dan perubahan keluarga.

Tugas 1 [Home](#) [Menu Penduduk](#) [Menu Keluarga](#) [Cari Penduduk](#)

Tambah Keluarga

Cari Keluarga

Ubah Keluarga

Fitur ini memudahkan ketika ingin mencari penduduk dan mengubah data penduduk, dibandingkan dengan mengetik langsung pada url.

- Fitur ubah status kematian penduduk di view penduduk.

Lihat Data Penduduk - 3101010009180001

Ubah Status Kematian

NIK	3101010009180001
Nama	Rampes Sampurasun
Tempat/Tanggal Lahir	Bandung, 2010-09-18
Alamat	Jln. Sutami No. 781
RT/RW	069/069
Kelurahan/Desa	PULAU UNTUNG JAWA
Kecamatan	KEPULAUAN SERIBU SELATAN
Kota	KABUPATEN KEPULAUAN SERIBU
Golongan Darah	AB+
Agama	Muslim
Status Perkawinan	Kawin
Pekerjaan	pelajar
Kewarganegaraan	WNI
Status Kematian	Wafat

Fitur sederhana ini sebenarnya berfungsi agar tidak usah membuat tampilan pencarian keluarga yang ingin dihapus.

Penjelasan project

Pada project ini, penulis menggunakan konsep MVC (model, view, controller). Penulis membuat 5 model yang terdiri dari, penduduk, kota, keluarga, kecamatan, dan kelurahan. Penulis juga membuat 2 controller untuk penduduk dan keluarga, serta 2 mapper untuk penduduk dan keluarga. Untuk bagian front end, penulis membuat folder template yang berisi file-file html untuk menampilkan halaman, folder statis yang berisi file javascript, css, dan bootstrap untuk desain, dan folder fragment yang berisi potongan-potongan html yang akan digunakan terus menerus. Cara kerja dari MVC yang penulis gunakan adalah seperti berikut. Mapper melakukan query untuk mengakses database dan menyimpannya ke model yang telah dibuat. Penulis membuat service untuk mengakses method pada mapper. Penulis juga membuat controller untuk memanggil service-service yang telah penulis buat dan mengarahkannya ke sebuah url.

Optimasi Database

Optimasi database yang penulis lakukan adalah:

- Penambahan primary key dan sifat unique ke tiap table. Dan juga menambahkan foreign key untuk table yang merefer ke table lain.
- Penambahan auto-increment ke table yang berfungsi untuk melakukan insertion, yaitu table penduduk dan keluarga.
- Penggunaan join untuk mengambil data dari 2 tabel yang berbeda dan saling merefer.

Stress testing dengan menggunakan JMeter

Pengujian akan dilakukan dengan Jmeter dengan kondisi *Number of Threads* sebesar 100 dan *rump up period* sebanyak 10.

Percobaan saat mencari data yang ada pada database: penduduk/view/?nik=3101019002020001

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: Browse... Log/Display Only: ☐ Errors ☐ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(...)	Status	Bytes	Sent Bytes	Latency	Connect Time(...)
1	22:50:07.330	Thread Group ...	HTTP Request	2888	✓	1074	164	2888	0
2	22:50:10.218	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
3	22:50:07.231	Thread Group ...	HTTP Request	2987	✓	1074	164	2987	2
4	22:50:10.221	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
5	22:50:07.431	Thread Group ...	HTTP Request	2790	✓	1074	164	2790	1
6	22:50:10.222	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
7	22:50:07.631	Thread Group ...	HTTP Request	7436	✓	1074	164	7436	0
8	22:50:07.531	Thread Group ...	HTTP Request	7536	✓	1074	164	7536	0
9	22:50:07.731	Thread Group ...	HTTP Request	7337	✓	1074	164	7337	0
10	22:50:15.068	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
11	22:50:15.068	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
12	22:50:15.068	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
13	22:50:07.832	Thread Group ...	HTTP Request	10791	✓	1074	164	10791	1
14	22:50:18.623	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
15	22:50:08.132	Thread Group ...	HTTP Request	10493	✓	1074	164	10493	1
16	22:50:18.626	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
17	22:50:08.032	Thread Group ...	HTTP Request	10612	✓	1074	164	10612	1
18	22:50:18.644	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
19	22:50:07.932	Thread Group ...	HTTP Request	10712	✓	1074	164	10712	0
20	22:50:18.644	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
21	22:50:08.333	Thread Group ...	HTTP Request	10316	✓	1074	164	10316	1
22	22:50:18.649	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
23	22:50:08.232	Thread Group ...	HTTP Request	10459	✓	1074	164	10458	0
24	22:50:18.693	Thread Group ...	HTTP Request	0	✗	1181	0	0	0
25	22:50:09.236	Thread Group ...	HTTP Request	13365	✓	1074	164	13365	1
26	22:50:09.535	Thread Group ...	HTTP Request	13066	✓	1074	164	13066	0
27	22:50:22.601	Thread Group ...	HTTP Request	0	✗	1181	0	0	0

☐ Scroll automatically? ☐ Child samples? No of Samples 200 Latest Sample 0 Average 8847 Deviation 9358

Pada hasil Jmeter di atas, dapat disimpulkan bahwa untuk melakukan proses pencarian penduduk sudah cukup baik. Tapi butuh banyak perbaikan. Average dari pencarian tersebut adalah 8847. Untuk fitur lainnya, ukuran average pada Jmeter masih tergolong cepat. Walaupun sudah melakukan optimasi pada database, waktu yang dibutuhkan untuk mengambil data yang banyak membutuhkan waktu yang cukup lama. Maka dibutuhkan perbaikan untuk algoritma pengambilan/pengkasesan data.