

OPTIMISASI DATABASE

Optimisasi database dilakukan dengan menggunakan Primary Index dengan Primary Key, serta Secondary Index dengan Foreign Key dan beberapa attribut yang sering digunakan sebagai Dense Index. Rincian dari index yang dibuat adalah sebagai berikut:

- Tabel Penduduk
 - Primary Index = id
 - Secondary Index
 - Foreign Key = id_keluarga -> keluarga (id)
 - Dense Index = nik, tanggal_lahir
- Tabel Keluarga
 - Primary Index = id
 - Secondary Index
 - Foreign Key = id_kelurahan -> kelurahan (id)
 - Dense Index = nomor_kk
- Tabel Kelurahan
 - Primary Index = id
 - Secondary Index
 - Foreign Key = id_kecamatan -> kecamatan (id)
- Tabel Kecamatan
 - Primary Index = id
 - Secondary Index
 - Foreign Key = id_kota -> kota (id)
- Tabel Kota
 - Primary Index = id

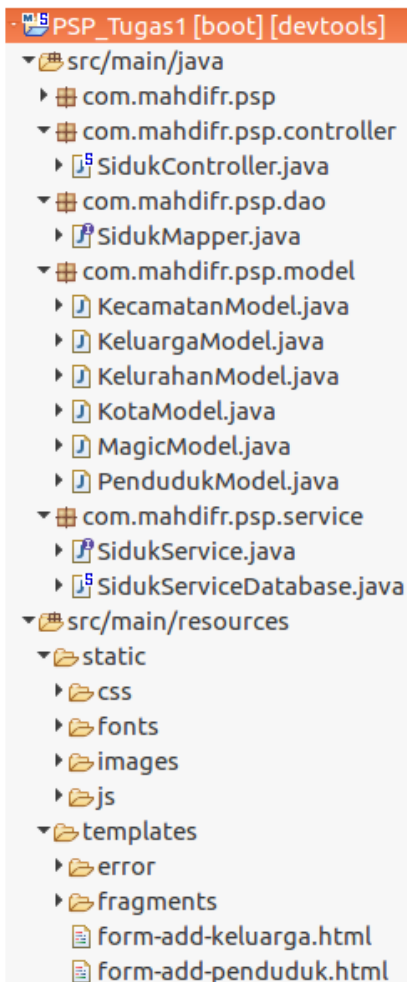
KONSEP

Sistem Informasi Kependudukan dibuat dengan konsep MVC dengan menggunakan framework Spring-boot dan menggunakan library DevTools, MySQL, Lombok, Thymeleaf, MyBatis, dan Web.

Implementasi ini digunakan untuk mendapatkan skalabilitas yang tinggi. Selain itu juga untuk mendapatkan keuntungan dari segi modularitas karena terbagi menjadi tiga layer. Akan tetapi, dikarenakan banyaknya dependency pada library lain, pengembangan dengan Spring-boot seringkali terkendala environment yang sulit untuk di tuning.

Library MyBatis yang digunakan dengan anotasi `@Select`, `@Delete`, `@Update` yang diajarkan dikelas juga berbeda dengan research yang saya lakukan terkait hubungan aplikasi dengan database. Konsep yang digunakan sangat sulit diimplementasikan untuk mengoptimalkan kinerja database. Hal ini dikarenakan untuk melakukan join pada `@Select`, Model yang menjadi return type harus memiliki seluruh attributes hasil join (walaupun ada cara menghandlenya dengan menggunakan struktur data `Map<k,v>`). Selain itu karena handle permasalahan tersebut adalah dengan chaining, maka query yang dilakukan sangatlah tidak efisien.

STRUKTUR PROJECT



Struktur project yang saya gunakan adalah dengan membagi package sesuai dengan layernya, yaitu Controller, Data Access Object (DAO), Model, dan Services. Selain itu juga untuk asset resource, terbagi menjadi dua, yaitu static dan template. Asset static dalam folder CSS berisi CSS Bootstrap dan DataTables, folder fonts berisi font dari Bootstrap, folder images berisi image untuk ascending dan descending DataTables, serta folder js berisi javascript dari Bootstrap, DataTables, dan javascript untuk dependency dropdown daerah.

Controller, DAO, serta Service yang dibuat adalah centralize, dibuat hanya satu pada masing-masing layer untuk seluruh aplikasi. Hal tersebut dikarenakan pertimbangan saya bahwa aplikasi ini masih sangat bergantung antar fiturnya, atau biasa saya sebut masih sebagai satu modul.

IMPLEMENTASI FITUR

■ Fitur 1: Mencari Penduduk berdasarkan NIK

Dalam implementasi fitur ini, pada controller saya menghandle nik imputan user, apakah sesuai dengan regex `\d*` atau tidak. Hal ini ditujukan untuk menghindari query injection. Setelah itu, dilakukan pengecekan apakah nik tersebut ada pada database atau tidak, jika ada maka akan ditampilkan detail dari penduduk tersebut, jika tidak maka akan muncul pesan error pada form tersebut.

```
/*
 * Fitur 1
 * @param String NIK
 */
@RequestMapping(value="/penduduk", method=RequestMethod.GET)
public String selectPenduduk(@RequestParam(value="nik", required=true) String nik, Model model) {
    if(nik.matches("\\d*")) {
        PendudukModel archive = sidukDAO.getBottomUpPenduduk(nik);
        if(archive == null) {
            model.addAttribute("errorNik", "NIK tidak ditemukan");
            return "home";
        } else {
            archive.setTanggal_lahir(reFormatStringYMDtoDMY(archive.getTanggal_lahir()));
            model.addAttribute("archive", archive);
            return "view-penduduk";
        }
    } else {
        model.addAttribute("errorNik", "Harus berisi angka");
        return "home";
    }
}
```

Error message jika tidak sesuai dengan regex:

SIDUK Home Add Penduduk Add Keluarga Search Penduduk

Lihat Data Penduduk Berdasarkan NIK

Masukan Nomor Induk Kependudukan

 Harus berisi angka

Lihat

Error jika tidak ada NIK tersebut pada database:

SIDUK Home Add Penduduk Add Keluarga Search Penduduk

Lihat Data Penduduk Berdasarkan NIK

Masukan Nomor Induk Kependudukan

 NIK tidak ditemukan

Lihat

Adapun tampilan jika data berhasil ditemukan adalah sebagai berikut:

SIDUK Home Add Penduduk Add Keluarga Search Penduduk

Lihat Data Penduduk - 3101011405170001

NIK = 3101011405170001

Nama = Heru Haryanto

Tempat/Tanggal Lahir = Jakarta 14-05-2017

Alamat = Ds. Adisumarmo No. 43

RT/RW = 079/025

Kelurahan/Desa = PULAU TIDUNG

Kecamatan = KEPULAUAN SERIBU SELATAN

Kota = KABUPATEN KEPULAUAN SERIBU

Golongan Darah = O-

Agama = Islam

Status Perkawinan = Belum Kawin

Pekerjaan = BELUM/TIDAK BEKERJA

Kewarganegaraan = WNI

Status Kematian = Hidup

Set Wafat

Adapun untuk menampilkan data tersebut, saya melakukan chaining query pada mapper. Hal ini sangatlah tidak efisien seperti yang telah dijelaskan pada bagian konsep.

```
@Select("SELECT * FROM kota WHERE id=#{id}")
KotaModel selectKota(@Param("id") String id);

@Select("SELECT * FROM kecamatan WHERE id=#{id}")
@Results(value = {
    @Result(property="id", column="id"),
    @Result(property="kode_kecamatan", column="kode_kecamatan"),
    @Result(property="nama_kecamatan", column="nama_kecamatan"),
    @Result(property="kota", column="id_kota",|
        javaType = KotaModel.class,
        many=@Many(select="selectKota"))
})
KecamatanModel selectKecamatan(@Param("id") String id);

@Select("SELECT * FROM kelurahan WHERE id=#{id}")
@Results(value = {
    @Result(property="id", column="id"),
    @Result(property="kode_kelurahan", column="kode_kelurahan"),
    @Result(property="nama_kelurahan", column="nama_kelurahan"),
    @Result(property="kode_pos", column="kode_pos"),
    @Result(property="kecamatan", column="id_kecamatan",
        javaType = KecamatanModel.class,
        many=@Many(select="selectKecamatan"))
})
KelurahanModel selectKelurahan(@Param("id") String id);
```

■ Fitur 2: Mencari Keluarga berdasarkan NKK

Dalam implementasi fitur ini, saya juga melakukan matching regex seperti yang telah dilakukan pada fitur pertama. Hal ini ditujukan juga untuk menghindari query injection. Setelah itupun juga dilakukan pengecekan pada database apakah data keluarga dengan nkk tersebut tersedia atau tidak.

```
/*
 * Fitur 2
 * @param String NKK
 */
@RequestMapping(value="/keluarga", method=RequestMethod.GET)
public String selectKeluarga(@RequestParam(value="nkk", required=true) String nkk, Model model) {
    if(nkk.matches("\\d*")) {
        KeluargaModel archive = sidukDAO.getTopDownKeluarga(nkk);
        if(archive == null) {
            model.addAttribute("errorNkk", "NKK tidak ditemukan");
            return "home";
        } else {
            model.addAttribute("archive", archive);
            return "view-detail-keluarga";
        }
    } else {
        model.addAttribute("errorNkk", "Harus berisi angka");
        return "home";
    }
}
```

Error message jika tidak sesuai dengan regex:

SIDUK Home Add Penduduk Add Keluarga Search Penduduk

Lihat Data Penduduk Berdasarkan NIK

Masukan Nomor Induk Kependudukan

Lihat Data Keluarga Berdasarkan NKK

Masukan Nomor Kartu Keluarga

 Harus berisi angka

Error message jika data tidak ditemukan pada database:

SIDUK Home Add Penduduk Add Keluarga Search Penduduk

Lihat Data Penduduk Berdasarkan NIK

Masukan Nomor Induk Kependudukan

Lihat Data Keluarga Berdasarkan NKK

Masukan Nomor Kartu Keluarga

 NKK tidak ditemukan

Adapun tampilan jika data berhasil ditemukan adalah sebagai berikut:

SIDUK

Home

Add Penduduk

Add Keluarga

Search Penduduk

Search

Lihat Data Keluarga - 3101010101050001

NKK = 3101010101050001

Alamat = Jln. Cikapayang No. 594

RT/RW = 149/090

Kelurahan/Desa = PULAU PARI

Kecamatan = KEPULAUAN SERIBU SELATAN

Kota = KABUPATEN KEPULAUAN SERIBU

No.	Nama	NIK	Jenis Kelamin	Tempat Lahir	Tanggal Lahir	Agama	Pekerjaan	Status Perkawinan	Status Dalam Keluarga	Kewarganegaraan
1	Taswir Rajata	3101010303890001	Laki-laki	Jakarta	1989-03-03	Islam	KARYAWAN HONORER	Kawin	Kepala Keluarga	WNI
2	Mulyono Hutagalung S.Gz	310101011408150001	Laki-laki	Jakarta	2015-08-14	Islam	BELUM/TIDAK BEKERJA	Belum Kawin	Anak	WNI
3	Yuliana Wahyuni	3101016312160001	Perempuan	Jakarta	2016-12-23	Islam	BELUM/TIDAK BEKERJA	Belum Kawin	Anak	WNI
4	Ifa Novitasari	3101014712830001	Perempuan	Jakarta	1983-12-07	Islam	PETANI/PEKEBUN	Kawin	Istri	WNI
5	test ganteng ganti tempat	3101010609970001	Laki-laki	Rawadenok	1997-09-06	Islam	Mahasiswa	Kawin	Anak	WNI

Tampilan yang saya dapatkan tersebut juga merupakan chaining query, dan query yang saya lakukan lebih banyak melibatkan record pada database karena turut menampilkan daftar anggota keluarga tersebut.

```
/*
 * Top -> Down : Keluarga-Penduduk
 * Fitur 2
 */
@Select("SELECT * FROM keluarga WHERE nomor_kk=#{nkk}")
@Results(value = {
    @Result(property="id", column="id"),
    @Result(property="nomor_kk", column="nomor_kk"),
    @Result(property="alamat", column="alamat"),
    @Result(property="rt", column="rt"),
    @Result(property="rw", column="rw"),
    @Result(property="is_tidak_berlaku", column="is_tidak_berlaku"),
    @Result(property="kelurahan", column="id_kelurahan",
        javaType = KelurahanModel.class,
        many=@Many(select="selectKelurahan")),
    @Result(property="penduduk", column="id",
        javaType = List.class,
        many=@Many(select="selectListAnggotaKeluarga"))
})
KeluargaModel selectDetailKeluarga(@Param("nkk") String nkk);

@Select("SELECT * FROM penduduk WHERE id_keluarga=#{nkk}")
List<PendudukModel> selectListAnggotaKeluarga(@Param("nkk") String nkk);
```

■ Fitur 3: Menambahkan Penduduk

Implementasi pada fitur menambahkan penduduk saya lakukan dengan memanfaatkan banyak final attributs seperti jenis_kelamin, golongan_darah, agama, status_perkawinan, kewarganegaraan, serta status_dalam_keluarga. Keseluruhan data dropdown tersebut saya rangkum berdasarkan database yang sudah tersedia.

```
// Initiate References Attributes
private final List<String> GOLONGAN_DARAH = Arrays.asList("A-", "A+", "B-", "B+", "AB-", "AB+", "O-", "O+");
private final List<String> AGAMA = Arrays.asList("Islam", "Kristen", "Katholik", "Hindu", "Budha", "Konghucu");
private final List<String> STATUS_PERKAWINAN = Arrays.asList("Kawin", "Belum Kawin", "Cerai Mati", "Cerai Hidup");
private final List<String> STATUS_DALAM_KELUARGA = Arrays.asList("Kepala Keluarga", "Istri", "Anak", "Famili Lain", "Pembantu");
```

SIDUK

Home

Add Penduduk

Add Keluarga

Search Penduduk

Tambah Penduduk

Masukan Nama

Tempat Lahir

Tanggal Lahir

Jenis Kelamin

Laki-laki

Golongan Darah

A-

Agama

Islam

Status Perkawinan

Kawin

Pekerjaan

Kewarganegaraan

WNI

Id Keluarga

Status Dalam Keluarga

Kepala Keluarga

Save

SIDUK

Home

Add Penduduk

Add Keluarga

Search Penduduk

Tambah Penduduk

Masukan Nama

Tempat Lahir

Tanggal Lahir

 Format dd-MM-yyyy

Laki-laki

A-

Islam

Kawin

WNI

Kepala Keluarga

Save

Pada masing-masing field saya lakukan validasi dengan menggunakan anotasi validasi pada model, seperti @NotNull, @Size, @Pattern, dan @Min. Lalu pada controller dibuatlah @Valid dan BindingResult seperti berikut:

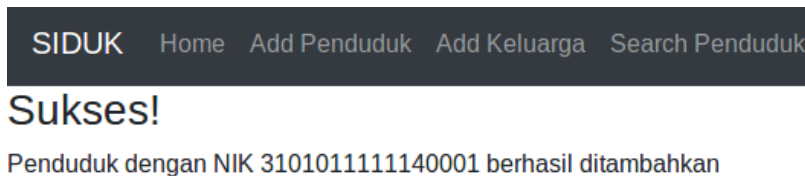
```

/*
 * Fitur 3
 * Insert Penduduk
 */
@RequestMapping(value="/penduduk/tambah", method=RequestMethod.GET)
public String formPenduduk(PendudukModel penduduk, Model model) {
    return "form-add-penduduk";
}

/*
 * @param PendudukModel
 */
@RequestMapping(value="/penduduk/tambah", method=RequestMethod.POST)
public String addPenduduk(@Valid PendudukModel penduduk, BindingResult bindingResult, Model model) {
    KeluargaModel archive = sidukDAO.getDataKeluarga(String.valueOf(penduduk.getId_keluarga()));
    if (archive == null) {
        model.addAttribute("errorIdKeluarga", "Id Keluarga Tidak Terdaftar");
        return "form-add-penduduk";
    } else {
        if (bindingResult.hasErrors()) {
            return "form-add-penduduk";
        } else {
            // Construct NIK
            String nik = constructNik(archive.getKelurahan().getKecamatan().getKode_kecamatan(),
                                     constructTanggal(penduduk.getTanggal_lahir()), penduduk.getJenis_kelamin());
            penduduk.setNik(nik);
            penduduk.setTanggal_lahir(reFormatStringDMYtoYMD(penduduk.getTanggal_lahir()));
            sidukDAO.insertPenduduk(penduduk);
            model.addAttribute("title", "Success Add Penduduk");
            model.addAttribute("message", "Penduduk dengan NIK " + nik + " berhasil ditambahkan");
            return "success-generic";
        }
    }
}
}

```

Adapun tampilan jika terdapat error pada pengisian form adalah seperti pada screenshot diatas. Dan jika penambahan penduduk berhasil dilakukan, maka akan muncul pesan sebagai berikut:



Untuk melakukan implementasi validasi tersebut, pada Model dibuat anotasi sebagai berikut:


```

1 package com.mahdifr.psp.model;
2
3 import javax.validation.constraints.NotNull;
4
5
6
7
8
9
10
11 @Data
12 @AllArgsConstructor
13 @NoArgsConstructor
14 public class PendudukModel {
15     private Integer id;
16     private String nik;
17
18     @NotNull(message = "Tidak Boleh Kosong")
19     private Integer jenis_kelamin, id_keluarga;
20
21     @NotNull(message = "Tidak Boleh Kosong")
22     @Size(min=1, max=128, message = "Minimum 1 Karakter, Maximum 128 Karakter")
23     private String nama, tempat_lahir, agama, pekerjaan, status_perkawinan, status_dalam_keluarga, golongan_darah;
24
25     @NotNull(message = "Tidak Boleh Kosong")
26     @Pattern(regexp = "\\d{2}(-|\\d{2}(-|\\d{4})", message = "Format dd-MM-yyyy")
27     private String tanggal_lahir;
28
29     @NotNull(message = "Tidak Boleh Kosong")
30     private boolean is_wni, is_wafat;

```

Dan pada mapper dibuatlah implementasi @insert untuk menambahkan penduduk tersebut:

```

/*
 * Fitur 3
 */
@Select("SELECT nik FROM penduduk WHERE nik BETWEEN #{minNik} AND #{maxNik}"
+ "ORDER BY nik DESC LIMIT 1")
String selectLastUrutanPenduduk(@Param("minNik") String minNik, @Param("maxNik") String maxNik);

@Insert("INSERT INTO penduduk (jenis_kelamin, id_keluarga, nik, nama, tempat_lahir, agama, pekerjaan, status_perkawinan, "
+ "status_dalam_keluarga, golongan_darah, tanggal_lahir, is_wni, is_wafat) VALUES (#{jenis_kelamin}, #{id_keluarga}, #{nik}, #{nama}, "
+ "#{tempat_lahir}, #{agama}, #{pekerjaan}, #{status_perkawinan}, #{status_dalam_keluarga}, #{golongan_darah}, #{tanggal_lahir}, "
+ "#{is_wni}, #{is_wafat})")
void insertPenduduk(PendudukModel penduduk);

```

Terkait pembuatan NIK, saya melakukan pengecekan pada range <NIK>0001 hingga <NIK>9999 dan melakukan order descending dan limit 1 sehingga mendapatkan NIK terakhir. Jika hasilnya null, maka akan digunakan <NIK>0001, sedangkan jika ada maka NIK yang digunakan adalah last NIK tersebut ditambah dengan 1.

Adapun helper method untuk melakukan construct NIK adalah sebagai berikut:

```

/*
 * Helper Method
 * Construct NIK
 */
private String constructNik(String kodeDaerah, String tanggal, int jenis_kelamin) {
    String newTanggal = "";
    if(jenis_kelamin == 1) {
        newTanggal = String.valueOf(Integer.parseInt(tanggal.substring(0,2))+40) + tanggal.substring(2, tanggal.length());
    } else {
        newTanggal = tanggal;
    }
    String minNik = kodeDaerah.substring(0, 6) + newTanggal.substring(0, 4) + newTanggal.substring(newTanggal.length()-2, newTanggal.length()) + "0001";
    String maxNik = String.valueOf(Long.parseLong(minNik)+9999);
    // Search nomor urut terakhir NIK
    String lastNoUrutNik = sidukDAO.getLastUrutanPenduduk(minNik, maxNik);
    if(lastNoUrutNik == null) {
        return minNik;
    } else {
        return String.valueOf(Long.parseLong(lastNoUrutNik)+1);
    }
}

```

■ Fitur 4: Menambahkan Keluarga

Dalam implementasi fitur ini, saya melakukan implementasi dependency dropdown untuk pemilihan domisili. Akan tetapi autofill pada dropdown daerah belum bisa diimplementasikan dengan baik dikarenakan kurangnya waktu yang saya alokasikan untuk mengembangkan fungsi tersebut. Terkait dependency dropdown akan saya jelaskan secara detail pada bagian fitur lain.

Pada controller, fitur ini diimplementasikan kurang lebih sama seperti menambahkan penduduk, yaitu sebagai berikut:

```
* Fitur 4
* Insert Keluarga
*/
@RequestMapping(value="/keluarga/tambah", method=RequestMethod.GET)
public String formKeluarga(KeluargaModel keluarga, Model model) {
    // Mengirimkan list kota sebagai initiate dependency dropdown
    List<KotaModel> listKota = sidukDAO.getListKota();
    model.addAttribute("kotaModel", listKota);
    return "form-add-keluarga";
}

/*
 * @param KeluargaModel
 */
@RequestMapping(value="/keluarga/tambah", method=RequestMethod.POST)
public String addKeluarga(@Valid KeluargaModel keluarga, BindingResult bindingResult, Model model) {
    if(keluarga.getId_kelurahan() == null) {
        List<KotaModel> listKota = sidukDAO.getListKota();
        model.addAttribute("kotaModel", listKota);
        model.addAttribute("errorRequired", "Harus Diisi");
        return "form-add-keluarga";
    } else {
        if(bindingResult.hasErrors()) {
            List<KotaModel> listKota = sidukDAO.getListKota();
            model.addAttribute("kotaModel", listKota);
            return "form-add-keluarga";
        } else {
            // Select kelurahan untuk kode kecamatan
            KelurahanModel archive = sidukDAO.getBottomUpKelurahan(String.valueOf(keluarga.getId_kelurahan()));
            Date currentDate = new Date();
            String stringDate = constructTanggal(currentDate);
            // Construct NKK
            String nkk = constructNkk(archive.getKecamatan().getKode_kecamatan(), stringDate);
            keluarga.setNomor_kk(nkk);
            sidukDAO.insertKeluarga(keluarga);
            model.addAttribute("title", "Success Add Keluarga");
            model.addAttribute("message", "Keluarga dengan NKK " + nkk + " berhasil ditambahkan");
            return "success-generic";
        }
    }
}
```

Adapun tampilan untuk form penambahan keluarga adalah sebagai berikut, beserta dengan error message ketika isi dari form tersebut tidak valid.

SIDUK Home Add Penduduk Add Keluarga

Tambah Keluarga

Alamat

RT

RW

Kota

Pilih Kota ▾

Kecamatan

Pilih Kecamatan ▾

Kelurahan/Desa

Pilih Kelurahan ▾

Save

SIDUK Home Add Penduduk Add Keluarga Search Penduduk

Tambah Keluarga

Alamat

 Minimum 1 Karakter, Maximum 256 Karakter

RT

 Minimum 1 Karakter, Maximum 3 Karakter

RW

 Minimum 1 Karakter, Maximum 3 Karakter

Kota

Pilih Kota ▾

 Harus Diisi

Kecamatan

Pilih Kecamatan ▾

 Harus Diisi

Kelurahan/Desa

Pilih Kelurahan ▾

 Harus Diisi

Save

Jika pengisian form dilakukan dengan benar, maka akan muncul tampilan informasi bahwa data tersebut telah diinput:

SIDUK Home Add Penduduk Add Keluarga Search Penduduk

Sukses!

Keluarga dengan NKK 3171022210170002 berhasil ditambahkan

Implementasi anotasi validasi pada Model juga kurang lebih sama seperti pada model Penduduk. Untuk melakukan construct NKK, prinsip yang digunakan juga sama dengan yang dilakukan pada penduduk. Dan pada mapper pun kurang lebih sama.

```

/*
 * Helper Method
 * Construct NKK
 */
private String constructNkk(String kodeDaerah, String tanggal) {
    String minNkk = kodeDaerah.substring(0, 6) + tanggal.substring(0, 4) + tanggal.substring(tanggal.length()-2, tanggal.length()) + "0001";
    String maxNkk = String.valueOf(Long.parseLong(minNkk)+9999);
    // Search nomor urut terakhir NKK
    String lastNoUrutNkk = sidukDAO.getLastUrutanKeluarga(minNkk, maxNkk);
    if(lastNoUrutNkk == null) {
        return minNkk;
    } else {
        return String.valueOf(Long.parseLong(lastNoUrutNkk)+1);
    }
}

```

Hal yang berbeda dengan implementasi tambah penduduk adalah digunakannya dependency dropdown dalam memilih domisili. Hal pertama yang dipilih adalah Kota, lalu mendapatkan list Kecamatan yang berada pada Kota tersebut. Selanjutnya setelah memilih Kecamatan, maka akan muncul list Kelurahan yang berada pada Kecamatan tersebut. Jika hal yang sebelumnya harus dipilih namun tidak dipilih, maka tidak akan ada pilihan untuk dropdown tersebut sebelum dropdown sebelumnya dipilih.

■ Fitur 5: Mengubah Penduduk

Implementasi pengubahan data penduduk dilakukan dengan menampilkan form pengisian penduduk namun dengan data yang sudah terisi. Pengisian form tersebut juga dilakukan validasi yang serupa agar data yang masuk adalah data yang valid, seperti pada contoh berikut:

SIDUK [Home](#) [Add Penduduk](#)

Masukan Nama

Ridwan Hasim Marpaung

Tempat Lahir

Atambua

Tanggal Lahir

01-01-1969

Jenis Kelamin

Laki-laki

Golongan Darah

A-

Agama

Islam

Status Perkawinan

Kawin

Pekerjaan

PEMBANTU RUMAH TANI

Kewarganegaraan

WNI

Id Keluarga

261

Status Dalam Keluarga

Pembantu

Update

SIDUK [Home](#) [Add Penduduk](#) [Add Keluarga](#) [Search Penduduk](#)

Masukan Nama

Ridwan Hasim Marpaung

Tempat Lahir

Minimum 1 Karakter, Maximum 128 Karakter

Tanggal Lahir

Format dd-MM-yyyy

Jenis Kelamin

Laki-laki

Golongan Darah

A-

Agama

Islam

Status Perkawinan

Kawin

Pekerjaan

PEMBANTU RUMAH TANI

Kewarganegaraan

WNA

Id Keluarga

Id Keluarga Tidak Terdaftar

Status Dalam Keluarga

Pembantu

Update

Jika pengisian form tersebut dilakukan dengan benar, maka data akan terupdate dan NIK akan berubah jika terjadi perubahan pada id_keluarga ataupun pada tanggal_lahir. Akan tetapi jika tidak ada perubahan pada kedua hal tersebut, maka NIK tidak akan diupdate.

Adapun tampilan jika berhasil melakukan update penduduk adalah sebagai berikut:

SIDUK [Home](#) [Add Penduduk](#) [Add Keluarga](#) [Search Penduduk](#)

Sukses!

Penduduk dengan NIK 3101010101690001 berhasil diubah

Untuk melakukan pengecekan update NIK seperti yang telah dijelaskan sebelumnya, dilakukanlah pengecekan sebagai berikut pada controller:

```
PendudukModel oldPenduduk = sidukDAO.getBottomUpPenduduk(nik);
String newNik = "";
// Update NIK jika tanggal lahir atau id keluarga berubah
if(!penduduk.getTanggal_lahir().equals(reFormatStringYMDtoDMY(oldPenduduk.getTanggal_lahir())) ||
    penduduk.getId_keluarga() != oldPenduduk.getId_keluarga()) {
    // Construct new NIK
    newNik = constructNik(archive.getKelurahan().getKecamatan().getKode_kecamatan(),
        constructTanggal(penduduk.getTanggal_lahir()), penduduk.getJenis_kelamin());
} else {
    newNik = nik;
}
```

Adapun pada mapper dilakukan implementasi sebagai berikut:

```
/*
 * Fitur 5
 */
@Update("UPDATE penduduk SET nik=#{nik}, nama=#{nama}, tempat_lahir=#{tempat_lahir}, "
    + "tanggal_lahir=#{tanggal_lahir}, jenis_kelamin=#{jenis_kelamin}, is_wni=#{is_wni}, "
    + "id_keluarga=#{id_keluarga}, agama=#{agama}, pekerjaan=#{pekerjaan}, status_perkawinan=#{status_perkawinan}, "
    + "status_dalam_keluarga=#{status_dalam_keluarga}, golongan_darah=#{golongan_darah}, is_wafat=#{is_wafat} "
    + "WHERE id=#{id}")
void updatePenduduk(PendudukModel penduduk);
```

■ Fitur 6: Mengubah Keluarga

Implementasi fitur ini dilakukan kurang lebih sama dengan fitur sebelumnya, yaitu update penduduk. Akan tetapi pada autofill domisili, juga terdapat masalah yaitu ketika perubahan domisili tidak diisi penuh, seperti jika hanya dipilih Kota dan Kecamatan, tanpa memilih Kelurahan, maka akan kembali pada page tersebut dengan sudah terisi sesuai data yang ada di database, bukan data temporer yang baru saja diisikan.

Kesalahan ini juga berlaku untuk field lainnya dikarenakan terdapat dua object yaitu yang ada di database untuk digunakan sebagai autofill, dan ada pula object temporer pengisian update keluarga tersebut. Akan tetapi object temporer tersebut tidak bisa digunakan untuk autofill dikarenakan tidak memiliki data hingga nama kota untuk digunakan sebagai autofill. Sehingga data pada database digunakan kembali sebagai dasar autofill.

Untuk implementasinya, pada controller serta mapper dilakukan hal yang serupa dengan update Penduduk. Hal yang berbeda adalah pengiriman list Kota, Kecamatan, dan Kelurahan pada controller.

```
// Select kota, kecamatan, dan kelurahan untuk auto fill dependency dropdown
List<KotaModel> listKota = sidukDAO.getListKota();
List<KecamatanModel> listKecamatan = sidukDAO.getListKecamatan(String.valueOf(keluarga.getKelurahan().getKecamatan().getKota().getId()));
List<KelurahanModel> listKelurahan = sidukDAO.getListKelurahan(String.valueOf(keluarga.getKelurahan().getKecamatan().getId()));
model.addAttribute("keluargaModel", keluarga);
model.addAttribute("kotaModel", listKota);
model.addAttribute("kecamatanModel", listKecamatan);
model.addAttribute("kelurahanModel", listKelurahan);
return "form-update-keluarga";
```

Selain itu, hal yang berbeda dengan update penduduk adalah ketika domisili dari sebuah keluarga berubah, maka seluruh anggota keluarga tersebut juga berubah NIK nya, dikarenakan salah satu komponen NIK adalah domisili. Untuk melakukannya dilakukan:

```
// Update NIK penduduk jika id kelurahan berubah
if(keluarga.getId_kelurahan() != oldKeluarga.getId_kelurahan()) {
    KeluargaModel listAnggotaKeluarga = sidukDAO.getTopDownKeluarga(nkk);
    for(PendudukModel penduduk : listAnggotaKeluarga.getPenduduk()) {
        String newNik = constructNik(archive.getKecamatan().getKode_kecamatan(),
            constructTanggal(reFormatStringYMDtoDMY(penduduk.getTanggal_lahir()), penduduk.getJenis_kelamin()));
        penduduk.setNik(newNik);
        sidukDAO.updatePenduduk(penduduk);
    }
}
```

Akan tetapi juga terdapat kesalah pada implementasi update keluarga. Saya tidak mengecek terlebih dahulu tanggal update keluarga tersebut di database melainkan langsung melakukan update. Hal ini dikarenakan pada awalnya saya mengira bahwa data akan disimpan hingga waktu perubahan.

Dikarenakan hal tersebut, jika sebuah keluarga di update walaupun tanpa perubahan pada domisili, maka NKK akan terincrement karena hasil dari pengecekan nomor urut NKK tersebut adalah dirinya sendiri.

Adapun tampilan pada fitur ini adalah sebagai berikut:

SIDUK

Home

Add Penduduk

Add Keluarga

Ubah Keluarga

Alamat

Jln. Sutarto No. 643

RT

08

RW

08

Kota

KOTA JAKARTA PUSAT

Kecamatan

CEMPAKA PUTIH

Kelurahan/Desa

CEMPAKA PUTIH TIMUR

Update

SIDUK

Home

Add Penduduk

Add Keluarga

Ubah Keluarga

Alamat

Jln. Sutarto No. 643

RT

08

RW

08

Kota

KOTA JAKARTA PUSAT

Kecamatan

CEMPAKA PUTIH

Kelurahan/Desa

CEMPAKA PUTIH TIMUR

Update

Harus Diisi

Harus Diisi

Harus Diisi

Seperti yang telah dijelaskan sebelumnya, terdapat masalah pada autofill, sehingga ketika ada field yang tidak sesuai, error akan muncul tetapi field akan sudah terisi kembali sesuai dengan yang ada di database.

■ Fitur 7: Mengubah Status Kematian Penduduk

Pada implementasi fitur ini, hanya perlu dilakukan update terhadap atribut `is_wafat` seorang penduduk. Hal tersebut dilakukan dengan implementasi controller sebagai berikut:

```
/*
 * Fitur 7
 * Set Kematian
 * @param String NIK
 */
@RequestMapping(value="/penduduk/mati", method=RequestMethod.POST)
public String ubahStatusKematian(@RequestParam(value="nik") String nik) {
    sidukDAO.setWafat(nik);
    // Set Tidak berlaku jika sudah tidak ada anggota keluarga yang masih hidup
    PendudukModel penduduk = sidukDAO.getBottomUpPenduduk(nik);
    int countHidup = sidukDAO.countJumlahAnggotaKeluargaHidup(penduduk.getKeluarga().getNomor_kk());
    if(countHidup == 0) {
        sidukDAO.setTidakBerlaku(penduduk.getKeluarga().getNomor_kk());
    }
    return "redirect:/penduduk?nik=" + nik;
}
```

Pada controller juga dilakukan pengecekan jumlah anggota keluarga yang hidup pada keluarga penduduk tersebut, jika jumlahnya sudah sama dengan 0, maka status berlaku pada Keluarga tersebut diubah menjadi tidak berlaku. Adapun tombol untuk mengubah status kematian ini ada pada fitur pertama.

SIDUK Home Add Penduduk Add Keluarga Search Penduduk

Lihat Data Penduduk - 3101010505960001

NIK = 3101010505960001

Nama = Baktiadi Ridwan Damanik

Tempat/Tanggal Lahir = Jakarta 05-05-1996

Alamat = Ds. Laksamana No. 235

RT/RW = 018/044

Kelurahan/Desa = PULAU TIDUNG

Kecamatan = KEPULAUAN SERIBU SELATAN

Kota = KABUPATEN KEPULAUAN SERIBU

Golongan Darah = O+

Agama = Kristen

Status Perkawinan = Kawin

Pekerjaan = ANGGOTA MAHKAMAH KONSTITUSI

Kewarganegaraan = WNI

Status Kematian = Hidup

Set Wafat

Sedangkan implementasi mapper untuk mengetahui jumlah anggota keluarga yang masih hidup serta untuk mengubah status kematian adalah sebagai berikut:


```
/*
 * Fitur 7
 */
@Update("UPDATE penduduk SET is_wafat=1 WHERE nik=#{nik}")
void setWafat(@Param("nik") String nik);

@Select("SELECT count(is_wafat) FROM keluarga JOIN penduduk "
        + "ON (keluarga.id=penduduk.id_keluarga) WHERE keluarga.nomor_kk=#{nkk} AND penduduk.is_wafat=0")
int countJumlahAnggotaKeluargaHidup(@Param("nkk") String nkk);

@Update("UPDATE keluarga SET is_tidak_berlaku=1 WHERE nomor_kk=#{nkk}")
void setTidakBerlaku(@Param("nkk") String nkk);
```

■ Fitur 8: Menampilkan daftar Penduduk pada suatu Daerah

Untuk implementasi fitur ini, saya melakukannya dengan dependency dropdown, sehingga sedikit berbeda dengan langkah-langkah yang diminta pada soal. Saya melakukannya langsung pada satu buah page dan satu buah method pada Controller. Pada controller, implementasinya adalah sebagai berikut:

```
/*
 * Fitur 8
 * Search Penduduk di suatu daerah
 */
@RequestMapping(value="/penduduk/cari", method=RequestMethod.GET)
public String cariPenduduk(Model model,
    @RequestParam(value="kt") Optional<String> kt,
    @RequestParam(value="kc") Optional<String> kc,
    @RequestParam(value="kl") Optional<String> kl) {
    if(kt.isPresent()) {
        if(kc.isPresent()) {
            if(kl.isPresent()) {
                // Return data table
                List<PendudukModel> archive = sidukDAO.getListPendudukDaerah(kl.get());
                model.addAttribute("pendudukModel", archive);
                // Penduduk paling muda
                PendudukModel termuda = sidukDAO.getPendudukTermudaDaerah(kl.get());
                model.addAttribute("pendudukTermuda", termuda);
                // Penduduk paling tua
                PendudukModel tertua = sidukDAO.getPendudukTertuaDaerah(kl.get());
                model.addAttribute("pendudukTertua", tertua);
                return "view-penduduk-by-daerah";
            }
            // Tidak mendapat kiriman kelurahan
            // Autofill untuk yang telah diisi
            List<KotaModel> listKota = sidukDAO.getListKota();
            List<KecamatanModel> listKecamatan = sidukDAO.getListKecamatan(kt.get());
            List<KelurahanModel> listKelurahan = sidukDAO.getListKelurahan(kc.get());
            model.addAttribute("kotaModel", listKota);
            model.addAttribute("selectedKota", kt.get());
            model.addAttribute("kecamatanModel", listKecamatan);
            model.addAttribute("selectedKecamatan", kc.get());
            model.addAttribute("kelurahanModel", listKelurahan);
            model.addAttribute("errorKelurahan", "Harus Diisi");
            return "form-cari-penduduk";
        }
        // Tidak mendapat kiriman kecamatan
        // Autofill untuk yang telah diisi
        List<KotaModel> listKota = sidukDAO.getListKota();
        List<KecamatanModel> listKecamatan = sidukDAO.getListKecamatan(kt.get());
        model.addAttribute("kotaModel", listKota);
        model.addAttribute("selectedKota", kt.get());
        model.addAttribute("kecamatanModel", listKecamatan);
        model.addAttribute("errorKecamatan", "Harus Diisi");
        model.addAttribute("errorKelurahan", "Harus Diisi");
        return "form-cari-penduduk";
    } else {
        // Initial Page atau kirim tapi tidak diisi
        // Mengirimkan list kota sebagai initiate dependency dropdown
        List<KotaModel> listKota = sidukDAO.getListKota();
        model.addAttribute("kotaModel", listKota);
        return "form-cari-penduduk";
    }
}
```

Sedangkan untuk mapper, implementasinya dilakukan hanya dengan melakukan sebuah request untuk list penduduk sebuah daerah dengan menggunakan operasi join.

```
/*
 * Fitur 8
 */
@Select("SELECT p.nik, p.nama, p.jenis_kelamin FROM penduduk p JOIN keluarga k ON p.id_keluarga=k.id "
+ "JOIN kelurahan kel ON k.id_kelurahan=kel.id WHERE kel.id=#{idKelurahan}")
List<PendudukModel> getListPendudukDaerah(@Param("idKelurahan") String idKelurahan);

@Select("SELECT p.nik, p.nama, p.tanggal_lahir FROM penduduk p JOIN keluarga k ON p.id_keluarga=k.id "
+ "JOIN kelurahan kel ON k.id_kelurahan=kel.id WHERE kel.id=#{idKelurahan} ORDER BY p.tanggal_lahir DESC LIMIT 1")
PendudukModel getPendudukTermudaDaerah(@Param("idKelurahan") String idKelurahan);

@Select("SELECT p.nik, p.nama, p.tanggal_lahir FROM penduduk p JOIN keluarga k ON p.id_keluarga=k.id "
+ "JOIN kelurahan kel ON k.id_kelurahan=kel.id WHERE kel.id=#{idKelurahan} ORDER BY p.tanggal_lahir ASC LIMIT 1")
PendudukModel getPendudukTertuaDaerah(@Param("idKelurahan") String idKelurahan);
```

Adapun tampilannya adalah sebagai berikut:

[SIDUK](#) [Home](#) [Add Penduduk](#) [Add Keluarga](#) [Search Penduduk](#)

Cari Penduduk

Kota

Pilih Kota ▼

Kecamatan

Pilih Kecamatan ▼

Kelurahan/Desa

Pilih Kelurahan ▼

Cari

[SIDUK](#) [Home](#) [Add Penduduk](#) [Add Keluarga](#)

Cari Penduduk

Kota

KOTA JAKARTA SELATAN ▼

Kecamatan

Pilih Kecamatan ▼ Harus Diisi

Kelurahan/Desa

Pilih Kelurahan ▼ Harus Diisi

Cari

Dan hasil setelah pencarian tersebut adalah sebagai berikut:

[SIDUK](#) [Home](#) [Add Penduduk](#) [Add Keluarga](#) [Search Penduduk](#)

Penduduk Dengan Usia Termuda

NIK 3171031909170001
Nama Lantar Habibi
Jenis Kelamin 2017-09-19

Penduduk Dengan Usia Tertua

NIK 3171034402530001
Nama Uli Yuniar M.Ak
Jenis Kelamin 1953-02-04

Show entries

No.	NIK	Nama	Jenis Kelamin	Detail Penduduk
1	3171035804940002	Pia Ilsa Wijayanti	Perempuan	<input type="button" value="Lihat detail"/>
2	3171036112640001	Zizi Usamah	Perempuan	<input type="button" value="Lihat detail"/>
3	3171036307150001	Maimunah Halimah	Perempuan	<input type="button" value="Lihat detail"/>
4	3171032907890001	Cahyadi Nainggolan	Laki-laki	<input type="button" value="Lihat detail"/>
5	3171032402150001	Bambang Mangunsong M.Farm	Laki-laki	<input type="button" value="Lihat detail"/>
6	3171036706130001	Jasmin Hartati	Perempuan	<input type="button" value="Lihat detail"/>
7	3171035108530001	Tari Kartika Agustina M.Pd	Perempuan	<input type="button" value="Lihat detail"/>
8	3171034204840001	Gabriella Handayani	Perempuan	<input type="button" value="Lihat detail"/>
9	3171030803870001	Dirja Saefullah	Laki-laki	<input type="button" value="Lihat detail"/>
10	3171030303100001	Hamzah Hasan Kurniawan	Laki-laki	<input type="button" value="Lihat detail"/>

FITUR TAMBAHAN

Dependency Dropdown

Implementasi fitur ini saya lakukan dengan menggunakan JQuery dan Ajax. Pada file javascript, dilakukan implementasinya sebagai berikut:

```
SidukController.java  view-penduduk.html  SidukMapper.java  PendudukModel.java  dependency-dropdown.js
1 $(document).ready(
2     function(){
3         $("#kota").change(
4             function() {
5                 $.getJSON("http://localhost:8080/beans/kecamatan", {
6                     id_kota : $(this).val(),
7                     ajax : 'true'
8                 }, function(data) {
9                     var htmlKecamatan = "<option disabled='true' selected='true' value=''>Pilih Kecamatan</option>";
10                    var htmlKelurahan = "<option disabled='true' selected='true' value=''>Pilih Kelurahan</option>";
11                    var lenKecamatan = data.kecamatan.length;
12                    for (var i = 0; i < lenKecamatan; i++) {
13                        htmlKecamatan += '<option value="' + data.kecamatan[i].id + ">'
14                        + data.kecamatan[i].nama_kecamatan + '</option>';
15                    }
16                    $("#kecamatan").html(htmlKecamatan);
17                    $("#kelurahan").html(htmlKelurahan);
18                });
19            }
20        );
21        $("#kecamatan").change(
22            function() {
23                $.getJSON("http://localhost:8080/beans/kelurahan", {
24                    id_kecamatan : $(this).val(),
25                    ajax : 'true'
26                }, function(data) {
27                    var htmlKelurahan = "<option disabled='true' selected='true' value=''>Pilih Kelurahan</option>";
28                    var lenKelurahan = data.kelurahan.length;
29                    for (var i = 0; i < lenKelurahan; i++) {
30                        htmlKelurahan += '<option value="' + data.kelurahan[i].id + ">'
31                        + data.kelurahan[i].nama_kelurahan + '</option>';
32                    }
33                    $("#kelurahan").html(htmlKelurahan);
34                });
35            }
36        );
37    }
38 );
```

Function dilakukan ketika ada perubahan pada dropdown Kota, yaitu akan dilakukan request ke Controller dengan menggunakan Ajax agar tampilan tidak berubah. Request tersebut dilakukan dengan meminta kembalian dari Controller berupa object JSON.

Ketika object tersebut diterima, maka akan dilakukan looping terhadap object tersebut untuk dilakukan pengisian terhadap dropdown Kecamatan. Hal tersebut juga berlaku untuk hubungan antara Kecamatan dan Kelurahan.

Adapun method yang digunakan pada controller untuk mengembalikan object JSON tersebut adalah sebagai berikut:

```

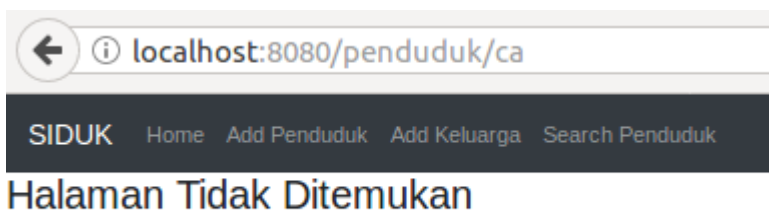
/*
 * Helper Method
 * Provide JSON untuk dependency dropdown
 */
@RequestMapping(value = "/beans/kecamatan", method = RequestMethod.GET)
public @ResponseBody KotaModel findListKecamatan(
    @RequestParam(value = "id_kota", required = true) String id_kota
) {
    return sidukDAO.getTopDownKota(id_kota);
}

@RequestMapping(value = "/beans/kelurahan", method = RequestMethod.GET)
public @ResponseBody KecamatanModel findListkelurahan(
    @RequestParam(value = "id_kecamatan", required = true) String id_kecamatan
) {
    return sidukDAO.getTopDownKecamatan(id_kecamatan);
}

```

Menampilkan Error Page

Seperti yang telah dijelaskan pada tutorial 6, implementasi error page ini dilakukan dengan cara yang sama, yaitu dengan membuat page error 404.html dan 500.html. Sehingga setiap kali terjadi error tersebut, page yang muncul adalah halaman tersebut:



Sedangkan error 500.html dapat terjadi salah satunya akibat kegagalan database.

Menampilkan Penduduk Paling Tua dan Paling Muda

Fitur tambahan ini sudah include kedalam fitur kedelapan, yaitu menampilkan penduduk termuda dan tertua pada suatu daerah:



Melihat Detail Penduduk dari Fitur 8: Menampilkan daftar Penduduk pada suatu Daerah

Selain penduduk dengan usia termuda dan tertua, pada fitur ke delapan juga ditampilkan sebuah tombol yang dika di klik maka akan melakukan redirect pada detail penduduk tersebut.

Show

10

 entries

Search:

No.	NIK	Nama	Jenis Kelamin	Detail Penduduk
1	3173034405810001	Lintang Ade Mardhiyah	Perempuan	<div>Lihat detail</div>
2	3173030109870001	Edi Gadang Dabukke	Laki-laki	<div>Lihat detail</div>
3	3173031108170001	Cakrawala Habibi	Laki-laki	<div>Lihat detail</div>
4	3173030801670001	Ade Opung Kurriawan	Laki-laki	<div>Lihat detail</div>
5	3173036704170001	Cinthia Kania Wastuti S.Kom	Perempuan	<div>Lihat detail</div>

STRESS TEST

Dalam melakukan stress test, terdapat tiga hal yang akan dilakukan secara keseluruhan, yaitu melakukan request detail data penduduk/keluarga, serta melakukan request daftar penduduk suatu daerah.

Pengujian yang dilakukan tidak menggunakan command line seperti yang direkomendasikan oleh Apache Jmeter, tetapi menggunakan GUI karena kekurangan kemampuan saya dalam mengoperasikannya tanpa GUI.

```
mahdifr@mahdifr-X550ZE:~/Downloads/apache-jmeter-3.2/bin$ java -jar Apache
JMeter.jar
=====
=====
Don't use GUI mode for load testing, only for Test creation and Test debug
ging !
For load testing, use NON GUI Mode:
    jmeter -n -t [jmx file] -l [results file] -e -o [Path to output folder]
& adapt Java Heap to your test requirements:
    Modify HEAP="-Xms512m -Xmx512m" in the JMeter batch file
=====
=====
```

Pengujian saya gunakan dengan menggunakan 10000 threads dengan ramp-up periods sebesar 50 untuk melihat detail data Penduduk/Keluarga. Angka tersebut saya gunakan dikarenakan jumlah warga DKI Jakarta yang juga banyak, selain itu juga dikarenakan keterbatasan laptop saya untuk melakukan pengujian. Adapun spesifikasi pengujian adalah sebagai berikut:

- CPU : AMD FX-7600P berjalan di 2.7 Ghz
- RAM : DDR3L 8GB 1600MHz
- Storage : HDD 1TB 5000Rpm

■ Detail data Penduduk/Keluarga

Apache JMeter (3.2 r1790748)

File Edit Search Run Options Help

0,0 KiB/s (0:41, 49%) 21:02

00:00:58 0 / 10000

Test Plan
Thread Group
HTTP Request
View Results in Table
WorkBench

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...
4	20:59:16.216	Thread Group...	HTTP Request	7	✓	2512	147	7	1
5	20:59:16.217	Thread Group...	HTTP Request	7	✓	2512	147	7	1
6	20:59:16.224	Thread Group...	HTTP Request	9	✓	2512	147	9	1
7	20:59:16.232	Thread Group...	HTTP Request	8	✓	2512	147	8	1
8	20:59:16.246	Thread Group...	HTTP Request	7	✓	2512	147	7	1
9	20:59:16.235	Thread Group...	HTTP Request	20	✓	2512	147	20	1
10	20:59:16.242	Thread Group...	HTTP Request	17	✓	2512	147	17	0
11	20:59:16.252	Thread Group...	HTTP Request	12	✓	2512	147	12	1
12	20:59:16.268	Thread Group...	HTTP Request	7	✓	2512	147	7	0
13	20:59:16.277	Thread Group...	HTTP Request	7	✓	2512	147	7	0
14	20:59:16.261	Thread Group...	HTTP Request	24	✓	2512	147	15	0
15	20:59:16.271	Thread Group...	HTTP Request	15	✓	2512	147	15	5
16	20:59:16.284	Thread Group...	HTTP Request	9	✓	2512	147	9	1
17	20:59:16.288	Thread Group...	HTTP Request	12	✓	2512	147	12	1
18	20:59:16.293	Thread Group...	HTTP Request	7	✓	2512	147	7	1
19	20:59:16.278	Thread Group...	HTTP Request	25	✓	2512	147	25	7
20	20:59:16.299	Thread Group...	HTTP Request	9	✓	2512	147	9	1
21	20:59:16.304	Thread Group...	HTTP Request	8	✓	2512	147	8	0
22	20:59:16.309	Thread Group...	HTTP Request	7	✓	2512	147	7	0
23	20:59:16.316	Thread Group...	HTTP Request	9	✓	2512	147	9	2
24	20:59:16.319	Thread Group...	HTTP Request	10	✓	2512	147	10	2
25	20:59:16.332	Thread Group...	HTTP Request	10	✓	2512	147	9	2
26	20:59:16.326	Thread Group...	HTTP Request	17	✓	2512	147	17	8
27	20:59:16.339	Thread Group...	HTTP Request	19	✓	2512	147	19	0
28	20:59:16.347	Thread Group...	HTTP Request	12	✓	2512	147	12	0
29	20:59:16.352	Thread Group...	HTTP Request	8	✓	2512	147	8	0
30	20:59:16.345	Thread Group...	HTTP Request	16	✓	2512	147	15	2
31	20:59:16.357	Thread Group...	HTTP Request	8	✓	2512	147	8	1
32	20:59:16.367	Thread Group...	HTTP Request	7	✓	2512	147	7	0
33	20:59:16.367	Thread Group...	HTTP Request	9	✓	2512	147	9	1

☐ Scroll automatically? ☐ Child samples? No of Samples 10000 Latest Sample 6 Average 15 Deviation 34

Apache JMeter (3.2 r1790748)

File Edit Search Run Options Help

0,0 KiB/s (0:39, 52%) 21:05

00:00:55 0 / 10000

Test Plan
Thread Group
HTTP Request
View Results in Table
WorkBench

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

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

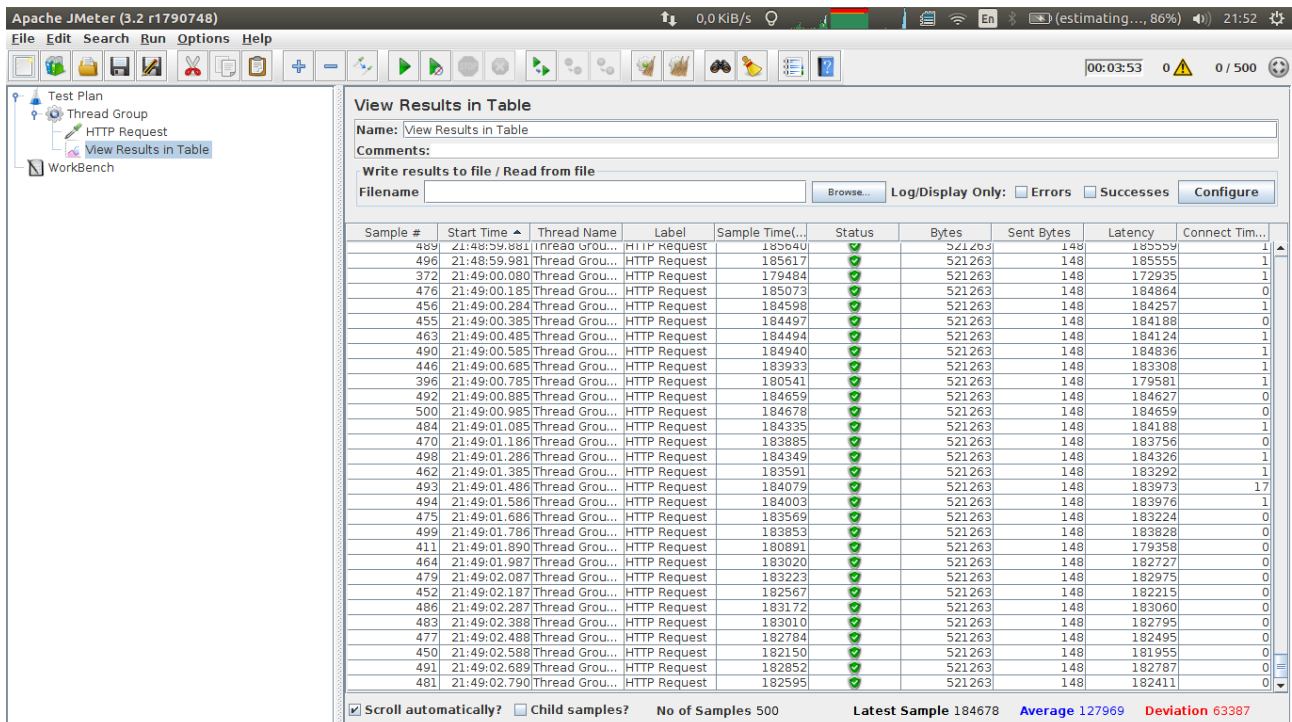
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Ti...
466	21:04:17.039	Thread Group...	HTTP Request	586	✓	4661	147	586	1
555	21:04:17.386	Thread Group...	HTTP Request	547	✓	4661	147	547	0
542	21:04:17.392	Thread Group...	HTTP Request	526	✓	4661	147	526	1
455	21:04:17.085	Thread Group...	HTTP Request	514	✓	4661	147	514	0
539	21:04:17.407	Thread Group...	HTTP Request	511	✓	4661	147	511	0
476	21:04:17.139	Thread Group...	HTTP Request	502	✓	4661	147	502	0
441	21:04:17.044	Thread Group...	HTTP Request	499	✓	4661	147	499	1
554	21:04:17.438	Thread Group...	HTTP Request	492	✓	4661	147	492	1
527	21:04:17.378	Thread Group...	HTTP Request	491	✓	4661	147	491	8
553	21:04:17.439	Thread Group...	HTTP Request	490	✓	4661	147	490	0
433	21:04:17.025	Thread Group...	HTTP Request	486	✓	4661	147	486	0
468	21:04:17.149	Thread Group...	HTTP Request	482	✓	4661	147	471	0
450	21:04:17.096	Thread Group...	HTTP Request	481	✓	4661	147	481	1
525	21:04:17.386	Thread Group...	HTTP Request	481	✓	4661	147	481	0
538	21:04:17.439	Thread Group...	HTTP Request	477	✓	4661	147	477	0
470	21:04:17.176	Thread Group...	HTTP Request	464	✓	4661	147	464	0
437	21:04:17.060	Thread Group...	HTTP Request	459	✓	4661	147	459	0
447	21:04:17.110	Thread Group...	HTTP Request	452	✓	4661	147	452	1
439	21:04:17.089	Thread Group...	HTTP Request	446	✓	4661	147	446	1
417	21:04:17.029	Thread Group...	HTTP Request	445	✓	4661	147	445	0
477	21:04:17.201	Thread Group...	HTTP Request	441	✓	4661	147	441	0
535	21:04:17.455	Thread Group...	HTTP Request	437	✓	4661	147	437	0
425	21:04:17.075	Thread Group...	HTTP Request	422	✓	4661	147	422	0
496	21:04:17.374	Thread Group...	HTTP Request	414	✓	4661	147	414	15
534	21:04:17.475	Thread Group...	HTTP Request	413	✓	4661	147	413	1
550	21:04:17.510	Thread Group...	HTTP Request	412	✓	4661	147	412	0
513	21:04:17.438	Thread Group...	HTTP Request	405	✓	4661	147	405	0
473	21:04:17.238	Thread Group...	HTTP Request	403	✓	4661	147	403	1
431	21:04:17.106	Thread Group...	HTTP Request	402	✓	4661	147	402	0
454	21:04:17.146	Thread Group...	HTTP Request	399	✓	4661	147	399	0

☐ Scroll automatically? ☐ Child samples? No of Samples 10000 Latest Sample 9 Average 18 Deviation 39

Untuk pengujian data Penduduk, rata-rata waktu akses adalah 15ms dengan standar deviasi sebesar 34. Sedangkan untuk pengujian data Keluarga, rata-ratanya adalah 18ms dengan standar deviasi sebesar 39. Dengan demikian, untuk sebuah laptop, hasil ini sangatlah baik. Jika diimplementasikan pada server, tentunya waktunya akan jauh lebih singkat.

■ Daftar Penduduk suatu Daerah

Dalam pengujian ini, saya melakukannya dengan menggunakan 500 Thread dengan Ramp-up Period sebesar 50. Hal tersebut menghasilkan rata-rata waktu akses sebesar 127.969ms dengan standar deviasi 63.387.



Apache JMeter (3.2 r1790749)

File Edit Search Run Options Help

0,0 KIB/s

00:03:53 0 0 / 500

Test Plan

- Thread Group
 - HTTP Request
 - View Results in Table
 - WorkBench

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 Tim...
489	21:48:59.881	Thread Group...	HTTP Request	185040	Success	521263	148	185559	1
496	21:48:59.981	Thread Group...	HTTP Request	185617	Success	521263	148	185555	1
372	21:49:00.080	Thread Group...	HTTP Request	179484	Success	521263	148	172935	1
476	21:49:00.185	Thread Group...	HTTP Request	185073	Success	521263	148	184864	0
456	21:49:00.284	Thread Group...	HTTP Request	184598	Success	521263	148	184257	1
455	21:49:00.385	Thread Group...	HTTP Request	184497	Success	521263	148	184188	0
463	21:49:00.485	Thread Group...	HTTP Request	184494	Success	521263	148	184124	1
490	21:49:00.585	Thread Group...	HTTP Request	184940	Success	521263	148	184836	1
446	21:49:00.685	Thread Group...	HTTP Request	183933	Success	521263	148	183308	1
396	21:49:00.785	Thread Group...	HTTP Request	180541	Success	521263	148	179581	1
492	21:49:00.885	Thread Group...	HTTP Request	184659	Success	521263	148	184627	0
500	21:49:00.985	Thread Group...	HTTP Request	184678	Success	521263	148	184659	0
484	21:49:01.085	Thread Group...	HTTP Request	184335	Success	521263	148	184188	1
470	21:49:01.186	Thread Group...	HTTP Request	183885	Success	521263	148	183756	0
498	21:49:01.286	Thread Group...	HTTP Request	184349	Success	521263	148	184326	1
462	21:49:01.385	Thread Group...	HTTP Request	183591	Success	521263	148	183292	1
493	21:49:01.486	Thread Group...	HTTP Request	184079	Success	521263	148	183973	17
494	21:49:01.586	Thread Group...	HTTP Request	184003	Success	521263	148	183976	1
475	21:49:01.686	Thread Group...	HTTP Request	183569	Success	521263	148	183224	0
499	21:49:01.786	Thread Group...	HTTP Request	183853	Success	521263	148	183828	0
411	21:49:01.890	Thread Group...	HTTP Request	180891	Success	521263	148	179358	0
464	21:49:01.987	Thread Group...	HTTP Request	183020	Success	521263	148	182727	0
479	21:49:02.087	Thread Group...	HTTP Request	183223	Success	521263	148	182975	0
452	21:49:02.187	Thread Group...	HTTP Request	182567	Success	521263	148	182215	0
486	21:49:02.287	Thread Group...	HTTP Request	183172	Success	521263	148	183060	0
483	21:49:02.388	Thread Group...	HTTP Request	183010	Success	521263	148	182795	0
477	21:49:02.488	Thread Group...	HTTP Request	182784	Success	521263	148	182495	0
450	21:49:02.588	Thread Group...	HTTP Request	182150	Success	521263	148	181955	0
491	21:49:02.689	Thread Group...	HTTP Request	182852	Success	521263	148	182787	0
481	21:49:02.790	Thread Group...	HTTP Request	182595	Success	521263	148	182411	0

☒ Scroll automatically? ☐ Child samples? No of Samples 500 Latest Sample 184678 Average 127.969 Deviation 63.387

Hal tersebut menunjukkan performa yang kurang baik, akan tetapi masih cukup reliable jika digunakan dengan spesifikasi server.