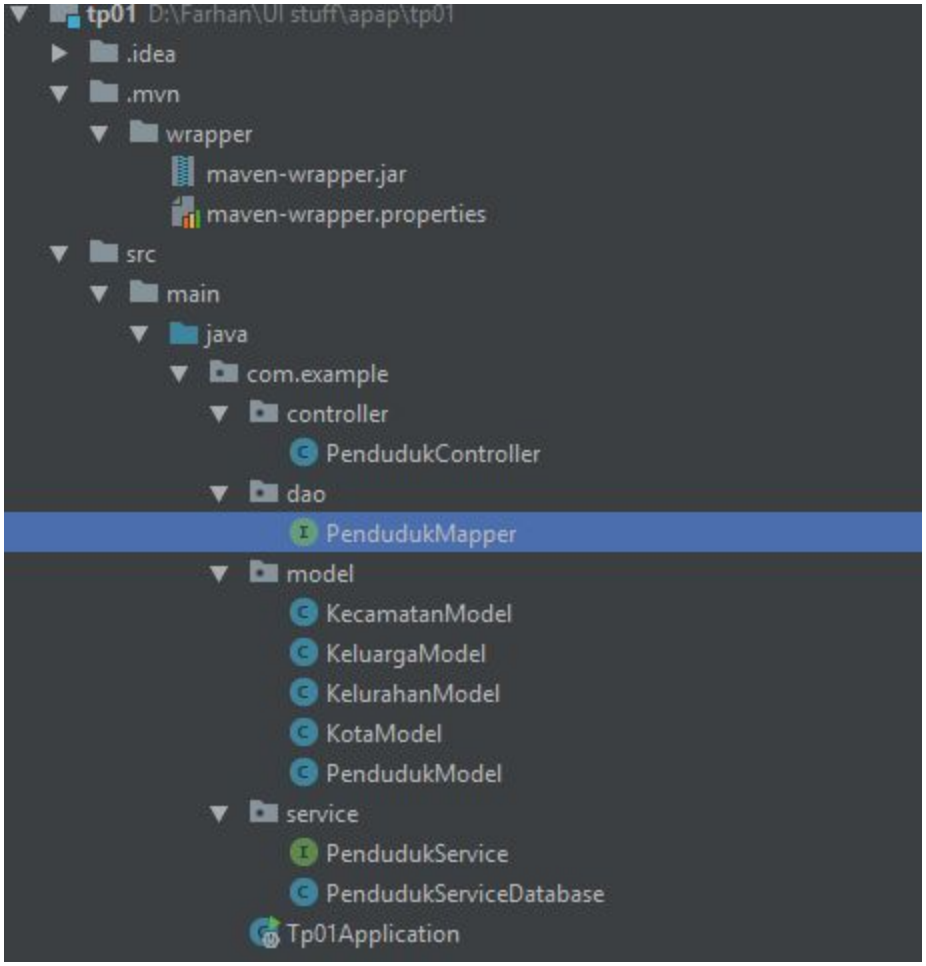
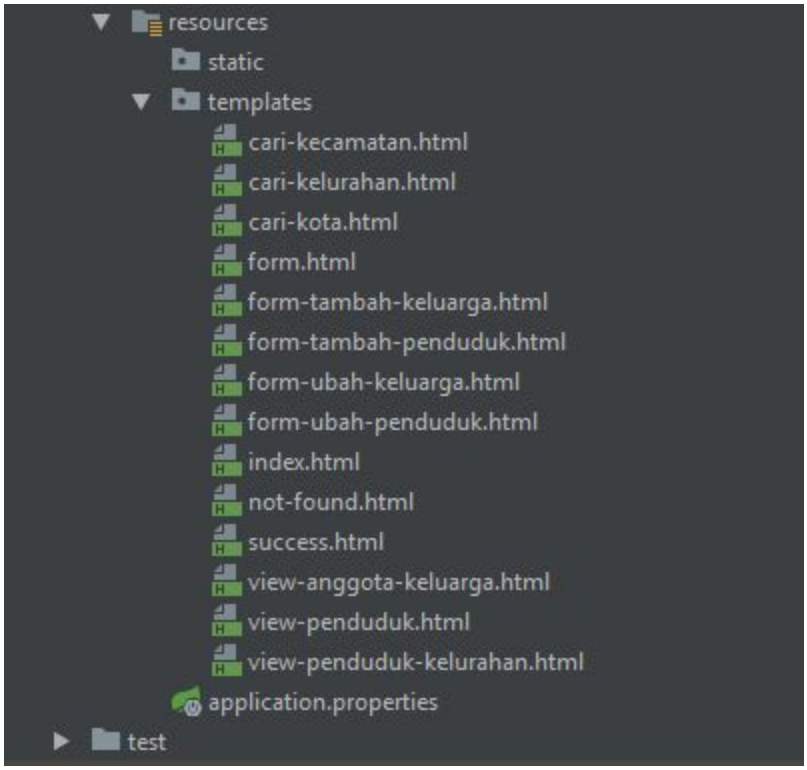
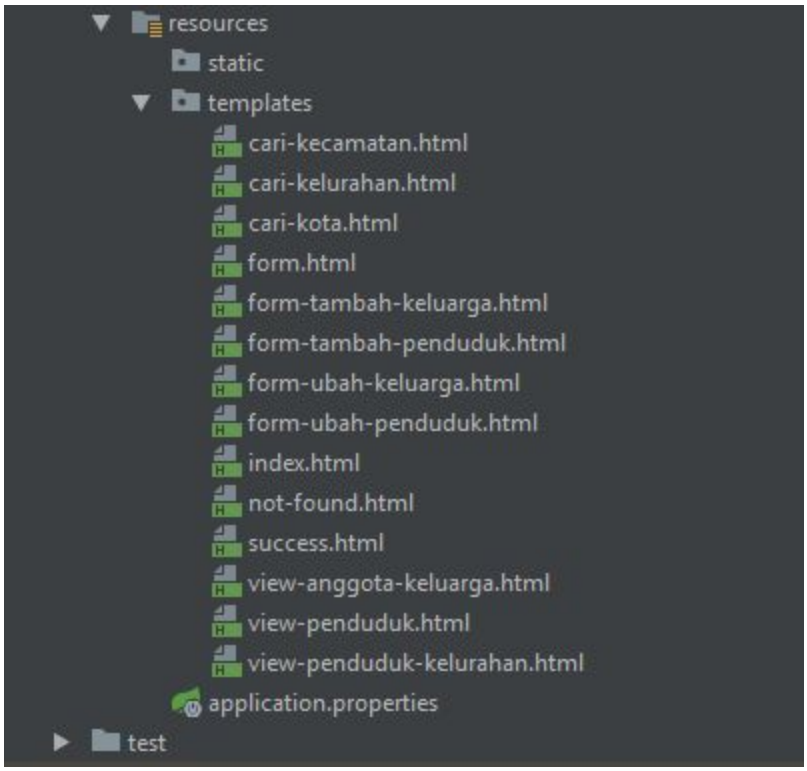
Pembuatan tp ini diawali dengan import database siduk_dki.sql dan pembuatan seperti model, service, template, mapper, controller, dll. Berikut skema projectnya:







Saya membuat 5 buah model untuk merepresentasikan kelima tabel yang ada pada database. Untuk isi dari tiap model, saya isi dengan column-name dari tabel dengan tambahan di masing-masing model yaitu:

1. Pada pendudukmodel saya menyimpan id_keluarga di dalam object keluargaModel
2. Pada keluargaModel saya menyimpan id_kelurahan di dalam object kelurahanModel dan juga list penduduk yang memiliki nomor kk yang sama(anggota keluarga)
3. Pada kelurahanModel saya menyimpan id_kecamatan di dalam object kecamatanModel dan juga list keluarga dengan id kelurahan yang sama
4. Pada kecamatanModel saya menyimpan id_kota di dalam object kotaModel dan juga list kelurahan dengan id kecamatan yang sama
5. Pada kotaModel saya menyimpan list kecamatan dengan id kota yang sama

Setelah itu mulailah saya mengisi pendudukController. Seluruh mapping akan difokuskan ke dalam pendudukController ini baik itu post, maupun get. Controller pertama kali yang saya isi adalah proses inisiasi pendudukDAO dan juga mapping index.

```
@Autowired
PendudukService pendudukDAO;

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

Jika program menerima path “/” maka akan mendirect ke form dimana user dapat mencari penduduk dan juga keluarga

Pengembangan fitur

Fitur 1:

Diawali dengan pembuatan controller

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

```java
@Mapper
public interface PendudukMapper {
    @Select("select nik, nama, tempat_lahir, tanggal_lahir, jenis_kelamin, is_wni, id_keluarga, agama, pekerjaan, status_perkawinan," +
            " status_dalam_keluarga, golongan_darah, is_wafat 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 = "is_wni", column = "is_wni"),
            @Result(property = "agama", column = "agama"),
            @Result(property = "pekerjaan", column = "pekerjaan"),
            @Result(property = "status_perkawinan", column = "status_perkawinan"),
            @Result(property = "status_dalam_keluarga", column = "status_dalam_keluarga"),
            @Result(property = "golongan_darah", column = "golongan_darah"),
            @Result(property = "is_wafat", column = "is_wafat"),
            @Result(property = "keluarga", column = "id_keluarga", javaType = KeluargaModel.class,  one = @One(select = "selectKeluarga"))

    })
    PendudukModel selectPenduduk(@Param("nik") String nik);
```

Yang mana akan mendirect ke template view-penduduk jika ketika selectPenduduk, terdapat penduduk yang dicari. Jika tidak ada maka akan didirect ke template not-found.

Pada template view-penduduk akan ditampilkan seluruh data yang terdapat pada database ditambah kelurahan, kecamatan, dan kota.

Fitur 2:
Diawali dengan pembuatan mapping pada controller

```java
@RequestMapping("/keluarga")
public String viewAnggotaKeluarga(Model model, @RequestParam(value = "nkk", required = false) String nkk){
    KeluargaModel keluarga = pendudukDAO.selectNKK(nkk);
    model.addAttribute( attributeName: "keluarga", keluarga);
    return "view-anggota-keluarga";
}
```

```java
@Select("select *, concat(rt, '/', rw) as rtrw from keluarga where keluarga.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_rw", column = "rtrw"),
        @Result(property = "anggota_keluarga", column = "id",
                javaType = List.class,
                many = @Many(select = "selectAnggotaKeluarga")),
        @Result(property = "kelurahan", column = "id_kelurahan",
                javaType = KelurahanModel.class,
                one = @One(select = "selectKelurahan"))
})
KeluargaModel selectNKK(@Param("nkk") String nkk);
```

Pada fitur ini,selain akan mencari data berkaitan dengan keluarga, akan dicari juga anggota keluarga pada saat selectNKK. Pada akhirnya akan ditampilkan semua data yang dicari yaitu data keluarga dan juga anggota keluarga.

Fitur 3:
Diawali dengan pembuatan mapping pada controller

```java
@RequestMapping("/penduduk/tambah")
public String tambahPenduduk(Model model){
    model.addAttribute( attributeName: "penduduk", new PendudukModel());
    return "form-tambah-penduduk";
}

@PostMapping ("/penduduk/tambah")
public String tambahPendudukSubmit(
        Model model,
        PendudukModel penduduk,
        @RequestParam(value = "tanggal_lahir") String tanggal_lahir,
        @RequestParam(value = "id_keluarga") String id_keluarga)
{
    String[] tokens = tanggal_lahir.split( regex: "-");
    String tanggal = tokens[2];
    if(penduduk.getJenis_kelamin() == 0){
       int tambah = parseInt(tokens[2].substring(0,1)) + 4;
       tanggal = Integer.toString(tambah) + tokens[2].substring(1,2);
    }
    String tanggallahir = tanggal +  tokens[1]  + tokens[0].substring(2);
    KeluargaModel keluarga = pendudukDAO.selectKeluarga(id_keluarga);
    String kode = keluarga.getKelurahan().getKecamatan().getKode_kecamatan().substring(0,6);
    String front = kode + tanggallahir;
    String nik = pendudukDAO.generateNIK(front, penduduk, id_keluarga);
    pendudukDAO.addPenduduk(penduduk, nik, id_keluarga);
    model.addAttribute( attributeName: "condition",  attributeValue: "tambah-penduduk");
    model.addAttribute( attributeName: "nik", nik);
    return "success";
}
```

```java
@Select("select * from penduduk where nik like #{front}")
List<PendudukModel> generateNIK(@Param("front") String front);
```

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

Pada controller akan dibentuk bagian depan depan dari nik yang terdiri dari kode lokasi dan tanggal lahir. Setelah itu, bagian belakang(pembeda nik) akan dibentuk menggunakan method generateNIK yang nantinya akan terbentuk keseluruhan nik. Setelah itu, peduduk akan dimasukkan ke dalam database dengan method addPenduduk.

Fitur 4:

```java
@RequestMapping ("/keluarga/tambah")
public String tambahKeluarga(Model model)
{
    model.addAttribute( attributeName: "keluarga", new KeluargaModel());
    return "form-tambah-keluarga";
}
@PostMapping("/keluarga/tambah")
public String tambahKeluargaSubmit(Model model, KeluargaModel keluarga,
                                   @RequestParam(value = "id_kelurahan") String id_kelurahan)
{
    KelurahanModel kelurahan = pendudukDAO.selectKelurahan(id_kelurahan);
    String kodeLokasi = kelurahan.getKecamatan().getKode_kecamatan().substring(0,6);
    String[] today = pendudukDAO.currentDate().split( regex: "-");
    String front = kodeLokasi + today[2] + today[1] + today[0].substring(2);
    String nkk = pendudukDAO.generateNKK(front, keluarga, id_kelurahan);
    model.addAttribute( attributeName: "condition",  attributeValue: "tambah-keluarga");
    model.addAttribute( attributeName: "nkk", nkk);
    return "success";
}
```

```java
@Select("SELECT CURDATE()")
String currentDate();

@Select("select * from keluarga where nomor_kk like #{front}")
List<KeluargaModel> generateNKK(@Param("front") String front);

@Insert("insert into keluarga(nomor_kk, alamat, RT, RW, id_kelurahan, is_tidak_berlaku) " +
        "values(#{nkk}, #{keluarga.alamat}, #{keluarga.rt}, #{keluarga.rw}, #{id_kelurahan}, 0)")
void addKeluarga(@Param("nkk") String nkk, @Param("keluarga") KeluargaModel keluarga, @Param("id_kelurahan") String id_kelurahan);
```

Pada fitur ini akan nkk akan dibentuk menggunakan method generateNKK. Setelah itu akan dimasukkan data keluarga ke dalam database dengan method generateNKK juga.

Fitur 5:

```java
}

@RequestMapping("/penduduk/ubah/{NIK}")
public String ubahPenduduk(Model model, @PathVariable("NIK") String nik){
    PendudukModel penduduk = pendudukDAO.selectPenduduk(nik);
    if(penduduk.getGolongan_darah().substring(1).equals("-")){
        penduduk.setGolongan_darah(penduduk.getGolongan_darah().substring(0,1)+"-");
    }
    model.addAttribute( attributeName: "penduduk", penduduk);
    return "form-ubah-penduduk";
}
```

```
@PostMapping("/penduduk/ubah/{NIK}")
public String ubahPendudukSubmit(Model model, PendudukModel penduduk,
                                 @RequestParam(value = "tanggal_lahir") String tanggal_lahir,
                                 @RequestParam(value = "id_keluarga") String id_keluarga)
{
    PendudukModel penduduk_lama = pendudukDAO.selectPenduduk(penduduk.getNik());
    if(!penduduk_lama.getTanggal_lahir().equals(penduduk.getTanggal_lahir())
            || !penduduk_lama.getKeluarga().getId().equals(id_keluarga)){

        // kondisi harus membuat NIK baru

        String[] tokens = tanggal_lahir.split( regex: "-");
        String tanggal = tokens[2];
        if(penduduk.getJenis_kelamin() == 0){
            int tambah = parseInt(tokens[2].substring(0,1)) + 4;
            tanggal = Integer.toString(tambah) + tokens[2].substring(1,2);
        }
        String tanggallahir = tanggal +  tokens[1]  + tokens[0].substring(2);
        KeluargaModel keluarga = pendudukDAO.selectKeluarga(id_keluarga);
        String kode = keluarga.getKelurahan().getKecamatan().getKode_kecamatan().substring(0,6);
        String front = kode + tanggallahir;
        String nik_baru = pendudukDAO.generateNIK(front, penduduk, id_keluarga);
        pendudukDAO.ubahPenduduk(penduduk, penduduk_lama.getNik(), nik_baru, id_keluarga);
    }
    else{
        pendudukDAO.ubahPenduduk(penduduk, penduduk_lama.getNik(), penduduk_lama.getNik(), id_keluarga);
    }
    model.addAttribute( attributeName: "condition",  attributeValue: "ubah-penduduk");
    model.addAttribute( attributeName: "nik", penduduk_lama.getNik());

    return "success";
```

Pada method ini akan diubahdata penduduk dengan NIK yang diminta. Selain itu juga akan mengecek apakah data yang berkaitan dengan NIK diganti atau tidak. Jika diganti akan dibentik NIK baru yang sesuai

Fitur 6:

```
@RequestMapping("/keluarga/ubah/{nkk}")
public String ubahKeluarga(Model model, @PathVariable("nkk") String nkk){
    KeluargaModel keluarga = pendudukDAO.selectNKK(nkk);
    model.addAttribute( attributeName: "keluarga", keluarga);
    return "form-ubah-keluarga";
}
```

```java
@RequestMapping("/keluarga/ubah/{nkk}")
public String ubahKeluarga(Model model, @PathVariable("nkk") String nkk){
    KeluargaModel keluarga = pendudukDAO.selectNKK(nkk);
    model.addAttribute( attributeName: "keluarga", keluarga);
    return "form-ubah-keluarga";
}

@PostMapping("/keluarga/ubah/{nkk}")
public String ubahKeluargaSubmit(Model model, KeluargaModel keluarga,
                                 @RequestParam(value = "id_kelurahan") String id_kelurahan)
{
    KeluargaModel keluarga_lama = pendudukDAO.selectNKK(keluarga.getNomor_kk());
    System.out.println(keluarga.getNomor_kk() + " kk lama");
    if(!keluarga_lama.getKelurahan().getId().equals(id_kelurahan))
    {
        // kondisi ketika NKK harus diganti

        KelurahanModel kelurahan = pendudukDAO.selectKelurahan(id_kelurahan);
        String kodeLokasi = kelurahan.getKecamatan().getKode_kecamatan().substring(0,6);
        String[] today = pendudukDAO.currentDate().split( regex: "-");
        String front = kodeLokasi + today[2] + today[1] + today[0].substring(2);
        String nkk_baru = pendudukDAO.generateNKK(front, keluarga, id_kelurahan);
        pendudukDAO.ubahKeluarga(keluarga_lama.getNomor_kk(), nkk_baru, keluarga, id_kelurahan);
    }
    else{
        pendudukDAO.ubahKeluarga(keluarga_lama.getNomor_kk(), keluarga_lama.getNomor_kk(), keluarga, id_kelurahan);
    }
    model.addAttribute( attributeName: "condition", attributeValue: "ubah-keluarga");
    model.addAttribute( attributeName: "nkk", keluarga_lama.getNomor_kk());

    return "success";
}
```

Pada method ini akan mengubah data keluarga dengan nomer kk yang bersangkutan. Pada mefitur ini juga jika terjadi perubahan pada data yang berhubungan dengan NKK, maka akan dibentuk NKK baru yang sesuai

Fitur 7:

```java
@PostMapping("/penduduk/mati")
public String nonaktifkanPenduduk(Model model,
                                  @RequestParam(value = "nik") String nik,
                                  @RequestParam(value = "nomor_kk") String nomor_kk){
    pendudukDAO.nonaktifkanPenduduk(nik);
    PendudukModel penduduk = pendudukDAO.selectPenduduk(nik);
    KeluargaModel keluarga = pendudukDAO.selectNKK(nomor_kk);
    int jumlah_keluarga = keluarga.getAnggota_keluarga().size();
    int i, is_wafat = 0;
    for(i = 0; i < jumlah_keluarga; i++){
        if(keluarga.getAnggota_keluarga().get(i).getIs_wafat() == 1){
            is_wafat=is_wafat + 1;
        }
    }
    System.out.println(is_wafat);
    if(jumlah_keluarga == is_wafat){
        pendudukDAO.nonaktifkanKeluarga(nomor_kk);
    }
    model.addAttribute( attributeName: "penduduk", penduduk);
    return "view-penduduk";
}
```

Pada method ini akan diubah nilai atribut is wafat dari penduduk menjadi mati. Selain itu juga akan dicek apakah keluarga yang bersangkutan valid atau tidak dengan menghitung jumlah anggota keluarga yang masih hidup.

Fitur 8:

```
@GetMapping(value = {"/penduduk/cari"})
public String cariPenduduk(Model model,
                           @RequestParam(value = "id_kota", required = false) String id_kota,
                           @RequestParam(value = "id_kecamatan", required = false) String id_kecamatan,
                           @RequestParam(value = "id_kelurahan", required = false) String id_kelurahan)
{
    if(!(id_kota == null)) {
        KotaModel kota = pendudukDAO.selectKota(id_kota);
        model.addAttribute( attributeName: "kecamatan_kecamatan", kota.getKecamatanKecamatan());
        model.addAttribute( attributeName: "id_kota", id_kota);
        model.addAttribute( attributeName: "nama_kota", kota.getNama_kota());
        if(!(id_kecamatan == null)){
            KecamatanModel kecamatan = pendudukDAO.selectKecamatan(id_kecamatan);
            model.addAttribute( attributeName: "kelurahan_kelurahan", kecamatan.getKelurahanKelurahan());
            model.addAttribute( attributeName: "id_kecamatan", id_kecamatan);
            model.addAttribute( attributeName: "nama_kecamatan", kecamatan.getNama_kecamatan());
            if(!(id_kelurahan == null)){
                System.out.print("masuk");
               model.addAttribute( attributeName: "id_kelurahan", id_kelurahan);
               List<PendudukModel> penduduk_penduduk = pendudukDAO.selectPendudukKelurahan(id_kelurahan);
               model.addAttribute( attributeName: "penduduk_penduduk", penduduk_penduduk);
               return "view-penduduk-kelurahan";
            }
            return "cari-kelurahan";
        }
        return "cari-kecamatan";
    }
```

```
else{
    List<KotaModel> kota_kota = pendudukDAO.cariKotakota();
    model.addAttribute( attributeName: "kota_kota", kota_kota);
    return "cari-kota";
}
```

Pada fitur ini akan dicari penduduk-penduduk yang berada pada kelurahan tertentu. Prosesnya adalah mencari kota dahulu, diikuti dengan kecamatan yang ada dalam kota tersebut, lalu terakhir kelurahan apa penduduk itu tinggal.

Untuk optimalisasi, saya menambahkan primary key pada tiap table dan juga menambahkan auto increment pada column id agar tiap penambahan tidak perlu mengisi id. Selain itu juga saya melakukan indexing pada table yang jarang diubah seperti kota,kecamatan, kelurahan agar proses pencarian berjalan dengan cepat. Pada stress testing cukup terjadi perubahan yang signifikan dengan perubahan diatas(index dan primary key).