**BAB III**

**PEMBAHASAN**

# KEBUTUHAN PERANGKAT LUNAK

Dalam pembuatan perangkat lunak pastinya diperlukan kebutuhan yang akan digunakan dalam pembangunan perangkat lunak itu sendiri, dan disini akan dibahas kebutuhan-kebutuhan yang akan digunakan dalam pembangunan perangkat lunak.

## Analisis Sistem Berjalan

1. Atasan memberi tahu karyawan untuk rapat secara manual;
2. Menggunakan sistem metting arranger lewat SMS gateway untuk memberi tahu rapat ke no masing-masing pegawai;



Gambar 14 Activity Diagram Sistem berjalan

## Analisis Arsitektur Sistem

Berikut ini merupakan gambar analisis arsitektur sitem aplikasi Info Bappeda



Gambar 15 Arsitektur Sistem

Gambaran arsitektur sistem yang akan dibangun memiliki dua aplikasi yang pertama yaitu aplikasi utama android dan yang kedua adalah aplikasi beckend atau sebagai web service. Penerima surat atau penegelola pemberitahuan rapat dapat menginputkan pengumuman rapat melalui web dashboard dan pegawai akan menerima pemberitahuan nya lewat aplikasi di android.

## Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak dibagi menjadi 2 kategori kebutuhan, yaitu kebutuhan fungsional dan kebutuhan non fungsional. Berikut ini kebutuhan fungsional dan kebutuhan non fungsional pada Info Bappeda:

### Kebutuhan Fungsional

Berikut ini merupakan tabel kebutuhan fungsional

Tabel 4 Kebutuhan Fungsional

|  |  |
| --- | --- |
| SRS-F-001 | Sistem dashboard menyediakan fasilitas bagi admin untuk menginputkan pemberitahuan rapat baru. |
| SRS-F-002 | Sistem dashboard menyediakan fasilitas melihat data pegawai |
| SRS-F-003 | Sistem android menyediakan fasilitas menampilkan list jadwal kegiatan rapat dan fasilitas notifikasi. |

### Kebutuhan Non Fungsional

Berikut ini merupakan tabel kebutuhan nonfungsional

Tabel 5 Kebutuhan Nonfungsional

|  |  |
| --- | --- |
| SRS-NF-001 | Sistem yang akan dibangun berbasis Mobile Android dan Website |
| SRS-NF-002 | Sistem yang akan dibangun akan mulus dijalankan minimal pada sistem operasi android Kitkat |
| SRS-NF-003 | Sistem yang akan dibangun memerlukan media penyimpanan minimal 20 MB untuk dapat menginstallnya |
| SRS-NF-004 | Sistem yang akan dibangun menggunakan bahasa pemrograman Java, Xml, php,json.html,javascript,css. |
| SRS-NF-005 | Sistem yang akan dibangun menggunakan DBMS Mysql |
| SRS-NF-006 | Sistem yang dibangun dengan spesifikasi hardware yang memenuhi standar minimum kebutuhan |
| SRS-NF-007 | Sistem yang akan dibangun akan memberikan notifikasi dalam bentuk suara dan getar |
| SRS-NF-008 | Pengguna yang akan menggunkan sistem ataupun aplikasi ini minimal mengerti mengoperasikan smartphone |
| SRS-NF-009 | Pengguna yang membutuhkan informasi mengenai jadwal rapat. |
| SRS-NF-010 | Sistem yang akan dibangun memerlukan koneksi internet |

## Analisis Kebutuhan Nonfungsional

Kebutuhan nonfungsional dibagi menjadi beberapa bagian, yaitu kebutuhan perangkat lunak, kebutuhan perangkat keras, kebutuhan perangkat pikir, dan kebutuhan jaringan. Berikut ini analisis kebutuhan nonfunsional pada Info Bappeda :

### Analisis Kebutuhan Perangkat Lunak

**Spesifikasi Kebutuhan NonFungsional Perangkat Lunak:**

Berikut ini merupakan tabel kebutuhan nonfungsional perangkat lunak

Tabel 6 kebutuhan Nonfungsional Perangkat lunak

|  |  |
| --- | --- |
| SRS-NF-001 | Sistem yang akan di bangun berbasis Mobile Android dan Website |
| SRS-NF-002 | Sistem yang akan di bangun akan mulus dijalankan minimal pada sistem operasi android Kitkat |
| SRS-NF-003 | Sistem yang akan di bangun memerlukan media penyimpanan minimal 20 MB untuk dapat menginstallnya |
| SRS-NF-004 | Sistem yang akan dibangun menggunakan bahasa pemrograman Java, Xml, php,json.html,javascript,css. |
| SRS-NF-005 | Sistem yang akan dibangun menggunakan DBMS MySql |

**Fakta perangkat Lunak (Yang ada dilingkungan sistem yang berjalan saat ini) :**

Sistem operasi : Android Kitkat.

Database : MySql

**Kebutuhan Pembangunan Perangkat Lunak :**

Dari sisi *developer*, perangkat lunak yang digunakan dalam pembangunan perangkat lunak ini adalah:

1. Platform : Mobile Android;
2. Sistem operasi : Android Kitkat;
3. Memory : Minimal 30 MB.
4. Bahasa pemrograman : Java, Xml, php,json.html,javascript,css. ;
5. DBMS : MySql;

**Kesimpulan (Hasil Perbandingan fakta dengan kebutuhan) :**

Kesimpulan antara kebutuhan perangkat lunak dan fakta yang ada di lingkungan sistem sudah memenuhi syarat kebutuhan.

### Analisis Kebutuhan Perangkat Keras

**Spesifikasi Kebutuhan NonFungsional Perangkat Keras:**

Tabel 7 kebutuhan nonfungsional Perangkat keras

|  |  |
| --- | --- |
| SRS-NF-006 | Sistem yang dibangun dengan spesifikasi hardware yang memenuhi standar minimum kebutuhan |
| SRS-NF-007 | Sistem yang akan di bangun akan memberikan notifikasi dalam bentuk suara dan getar |

**Fakta perangkat Keras (Yang ada dilingkungan sistem yang berjalan saat ini) :**

**Android**

Procesor : Snapdragon 435 1.0 GHz

Ram : 1 GB

Memory : 8 GB

Layar : 4.0’’

**Website**

Procesor : Intel Dual Core,i3,i5,i7.

Ram : 1 GB

Memory : 120 GB

Layar : 11’’

**Kebutuhan Perangkat Keras :**

**Android**

Platform : Mobile Android

Procesor : ARM Cortex-A7 1.0 GHz

Ram : 512 Mb

Memory : 4GB

Layar : 3.5’’

**Website**

Platform : Website

Procesor : Intel Core

Ram : 1 Mb

Memory : 120GB

Layar : 11’’

**Kesimpulan (Hasil Perbandingan fakta dengan kebutuhan) :**

Berdasarkan SKPL-NF-004 dan SKPL-NF-005 antara fakta dengan kebutuhan sistem sudah memenuhi syarat minimal.

### Analisis Kebutuhan Perangkat Pikir

Analisis ke butuhan Perangkat pikir yang dilakukan pada penelitian ini hanya mengenai costumer segment.

Tabel 8 Kebutuhan perangkat pikir

|  |  |  |  |
| --- | --- | --- | --- |
| **Costumer** | **Umur** | **Behavior** | **Psycologis** |
| Pegawai | 23-60 Tahun | Butuh informasi mengenai jadwal rapat. | Kondisinya Normal. |

**Kebutuhan Nonfungsional**

Berikut ini merupakan tabel kebutuhan Nonfungsional perangkat pikir

Tabel 9 Kebutuhan Nonfungsional

|  |  |
| --- | --- |
| SRS-NF-008 | Pengguna yang akan menggunkan sistem ataupun aplikasi ini minimal mengerti mengoperasikan smartphone |
| SRS-NF-009 | Pengguna yang membutuhkan informasi kegiatan jadwal rapat |

**Fakta Perangkat Pikir**

Berikut ini merupakan tabel fakta perangkat pikir

Tabel 10 Fakta Perangkat Pikir

|  |  |  |  |
| --- | --- | --- | --- |
| **Costumer** | **Umur** | **Behavior** | **Pengalaman menggunakan smartphone** |
| Pegawai | 23 Tahun | Tidak tahu jadwal rapat. | Expert |

**Kesimpulan (Hasil perbandingan Fakta dan kebutuhan)**

Berdasarkan SRS-NF-006 dan SRS-NF-007 fakta yang ada di lapangan sesuai dengan kebutuhan yang diperlukan oleh sistem.

### Analisis Kebutuhan Jaringan

**Kebutuhan Nonfongsional jaringan:**

Berikut ini merupakan tabel kebutuhan nonfungsional jaringan :

Tabel 11 Kebutuhan nonfungsional jaringan

|  |  |
| --- | --- |
| SRS-NF-010 | Sistem yang akan di bangun memerlukan koneksi internet |

**Fakta Jaringan (Yang ada di lingkungan sistem):**

Menggunakan MySql yang berfungsi untuk menampung data yang di inputkan, data yang berhasil di inputkan dari aplikasi website akan di tampilkan di halaman kegiatan aplikasi android dalam bentuk list data.

**Kesimpulan (Hasil Perbandingan fakta dengan kebutuhan) :**

Berdasarkan SKPL-NF-008 antara fakta dengan kebutuhan sistem sudah memenuhi syarat minimal.

## Analisis Kebutuhan Data

Analisis data yang menjadi acuan dalam membangun perangkat lunak ini adalah dokumen-dokumen yang berhubungan dengan donasi serta objek-objek lain yang terkait. Berikut adalah analisis data dari pembangunan perangkat lunak ini:

Tabel 12 Kebutuhan Data

| **ENTITAS** | **ATRIBUT** |
| --- | --- |
| t\_user | = {id\_user, nama,password,no\_hp,email,status,akses,nip,token} |
| t\_kegiatan | = {id\_kgiatan, nama,tempat,asalsurat,waktu\_mulai,waktu\_selesai,tanggal} |

Dari entitas tersebut, asumsi dalam penelitian ini adalah:

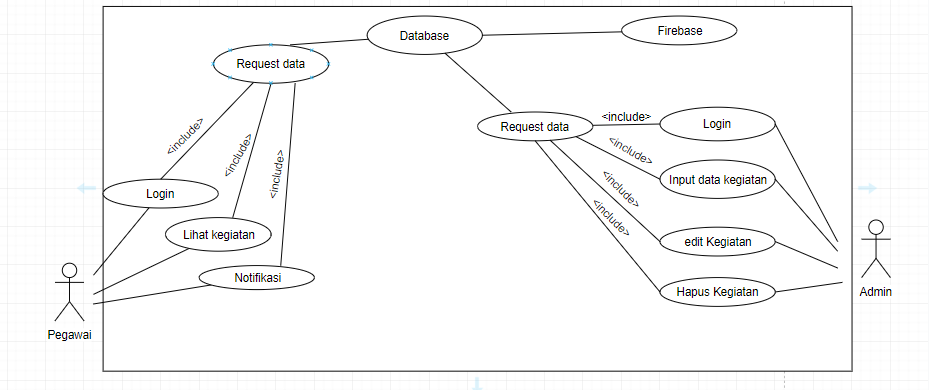
1. Entitas t\_user terdapat 2 login yaitu login admin dan login pegawai yang membedakan nya adalah field status admin/user.
2. Entitas t\_kegiatan menampung semua data kegiatan rapat yang di inputkkan admin.

Konsep analisis data yang digunakan adalah dengan menggunakan konsep ORM (*Object Relational Mapping*) dimana ORM ini akan memetakan sebuah objek dengan *database*. Dengan kata lain, ORM ini akan membantu menjadi jembatan antara objek yang didefinisikan dalam kode program dengan *database*. Objek-objek yang akan terlibat dalam ORM ini adalah objek recepsionis dan pegawai.

## Model Use Case

Pada bagian model *usecase* ini akan dipaparkan mengenai gambar diagram *usecase* keseluruhan, definisi aktor, definisi *usecase* dan skenario *usecase*.

### Diagram Use Case



Gambar 16 Usecase Diagram

### Definisi Actor

Berikut ini merupakan tabel definisi Actor

Tabel 13 Definisi aktor

| **NO** | **AKTOR** | **DESKRIPSI** |
| --- | --- | --- |
| *1* | *Admin (Humas)* | *Aktor dengan role ini mempunyai wewenang untuk melakukan penambahan list kegiatan (jadwal rapat), ubah list kegiatan dan hapus list kegiatan.* |
| *2* | *User(Pegawai)* | *Aktor dengan role ini mempunyai wewenang untukk login dan melihat list kegiatan serta menerima notifikasi jadwal rapat..* |

### Definisi Use Case

Berikut ini merupakan tabel definisi use case

Tabel 14 Definisi usecase

| **NO** | **USE CASE** | **DESKRIPSI** |
| --- | --- | --- |
| *1* | Input Kegiatan | *Admin menambah sebuah kegiatan rapat baru* |
| *2* | Ubah Jadwal Kegiatan rapat | *Admin mengubah jadwal kegiatan rapat dan sistem menampilkan form ubah list kegiatan* |
| *3* | Hapus Jadwal Kegiatan rapat | *Admin menghapus jadwal kegiatan rapat dan sistem menampilkan list kegiatan yang tersisa* |
| *4* | Notifikasi | *Sistem menampilkan Notifikasi dari Firebase* |
| *5* | Request data | *Sistem menampilkan daftar list kegiatan dari database* |
| *6* | Database | *Menampung data yang di requet oleh Actor* |
| *7* | Login | *Admin Login pada aplikasi Dasboard, pegawai login pada aplikasi Mobile android* |

### Skenario Use Case

Berikut ini merupakan beberapa tabel skenario use case :

Tabel 15 Usecase Scenario Tambah Data Rapat Aplikasi Dashboard

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Input Kegiatan Rapat | |
| **Related Requirement** | - | |
| Goal In Context | Menambahkan data kegiatan ke dalam aplikasi dashboard | |
| Description | Fungsionalitas ini digunakan oleh admin untuk menambahkan data kegiatan ke dalam database aplikasi | |
| Related Use Case | Request data kegiatan | |
| Preconditions | - | |
| Successful End Condition | Aktor berhasil menambahkan data kegiatan rapat | |
| Failed End Condition | Aktor tidak menambahkan data kegiatan rapat | |
| Primary Actors | User (Mahasiswa) | |
| Secondary Actors | - | |
| Trigger | User memilih/menekan tombol berwarna biru dan text “save” | |
| Main Flow | Step | Action |
|  | 1. | User memilih tombol berwarna merah dan berbentuk “Tambah Kegiatan” |
|  | 2. | Sistem menampilkan halaman penginputan data kegiatan |
|  | 3. | User mengisi data kegiatan berupa nama kegiatan beserta waktu dan tempat |
|  | 4.  include:: CRUD data kegiatan | Sistem menyimpan data kegiatan baru ke dalam database |
|  | 5. | Sistem menampilkan halaman home yang berisi list data kegiatan terbaru |
| Extension | Step | Branching Action |
|  | 4.1 | Sistem gagal menyimpan data kegiatan |
|  | 5.1 | Sistem gagal menampilkan halaman home yang berisi list data kegiatan terbaru |

Tabel 16 Usecase Scenario Ubah Kegiatan Rapat Aplikasi Dashboard

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Ubah Kegiatan rapat | |
| **Related Requirement** | - | |
| Goal In Context | Mengubah data kegiatan rapat yang sudah tersimpan | |
| Description | Fungsionalitas ini digunakan oleh admin untuk mengubah data kegiatan rapat dalam aplikasi dashboard | |
| Related Use Case | Request data kegiatan | |
| Preconditions | - | |
| Successful End Condition | Aktor berhasil mengubah data kegiatan rapat | |
| Failed End Condition | Aktor tidak mengubah data kegiatan rapat | |
| Primary Actors | User (Mahasiswa) | |
| Secondary Actors | - | |
| Trigger | User memilih/menekan kegiatan yang akan diubah | |
| Main Flow | Step | Action |
|  | 1. | Admin menekan button ubah pada kegiatan yang akan diubah |
|  | 2.  Include:: Request data kegiatan | Sistem melakukan permintaan detail data kegiatan ke database |
|  | 3. | Sistem menampilkan halaman detail data kegiatan |
|  | 4. | User mengubah detail data kegiatan. |
|  | 5.  Include:: CRUD data kegiatam | Sistem akan menyimpan data kegiatan yang sudah diubah ke database |
|  | 6. | Sistem menampilkan halaman home dashboard yang berisi list data kegiatan terbaru |
| Extension | Step | Branching Action |
|  | 2.1 | Sistem gagal mengambil detail data kegiatan rapat |
|  | 5.1 | Sistem gagal menyimpan detail data kegiatan rapat |
|  | 6.1 | Sistem gagal menampilkan halaman home yang berisi list data kegiatan terbaru |

Tabel 17 Usecase Scenario Hapus Jadwal Kegiatan Rapat Aplikasi Dashboard

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Hapus Kegiatan rapat | |
| **Related Requirement** | - | |
| Goal In Context | Menghapus data kegiatan rapat yang sudah tersimpan | |
| Description | Fungsionalitas ini digunakan oleh admin untuk menghapus data kegiatan rapat dalam aplikasi dashboard | |
| Related Use Case | Request data kegiatan | |
|  | - | |
| Successful End Condition | Aktor berhasil menghapus data kegiatan | |
| Failed End Condition | Aktor tidak menghapus data kegiatan | |
| Primary Actors | User (Admin) | |
| Secondary Actors | - | |
| Trigger | User memilih/menekan tombol berbentuk tong sampah | |
| Main Flow | Step | Action |
|  | 1. | User menekan kegiatan yang akan dihapus |
|  | 2. | Sistem menampilkan popup pesan “kegiatan akan di hapus?” |
|  | 3.  Include:: CRUD data kegiatam | Sistem akan menghapus data kegiatan rapat yang dipilih pada database |
|  | 6. | Sistem menampilkan halaman home yang berisi list data kegiatan terbaru |
| Extension | Step | Branching Action |
|  | 2.1 | Sistem gagal mengambil detail data kegiatan |
|  | **4.1** | **Sistem menampilkan validasi penghapusan** |
|  | 5.1 | Sistem gagal menyimpan detail data kegiatan rapat |
|  | 6.1 | Sistem gagal menampilkan halaman home yang berisi list data kegiatan terbaru |

Tabel 18 Usecase Scenario Lihat Jadwal Kegiatan Rapat Aplikasi Mobile

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Lihat Jadwal Kegiatan Rapat | |
| **Related Requirement** | - | |
| Goal In Context | User (Pegawai) dapat melihat daftar kegiatan rapat | |
| Description | Fungsionalitas ini digunakan oleh user untuk melihat jadwal rapat | |
| Related Use Case | - | |
| Preconditions | - | |
| Successful End Condition | Aktor berhasil melihat daftar kegiatan | |
| Failed End Condition | Aktor tidak dapat melihat kegiatan rapat | |
| Primary Actors | User (Pegawai) | |
| Secondary Actors | - | |
| Trigger | Aktor memilih/menekan tombol list kegiatan | |
| Main Flow | Step | Action |
|  | 1. | User memilih/menekan tombol kegiatan |
|  | 2. | User menekan list kegiatan yg dipilih untuk melihat detail kegiatan |
| Extension | Step | Branching Action |
|  | 2.1 | Daftar tidak tampil pada halaman kegiatan. |

Tabel 19 Usecase Scenario Notifikasi pada Aplikasi Mobile

|  |  |  |
| --- | --- | --- |
| **Use Case Name** | Notifikasi kegiatan baru | |
| **Related Requirement** | - | |
| Goal In Context | Menampilkan pesan notifikasi kegiatan baru | |
| Description | Memberitahu kegiatan baru | |
| Related Use Case | - | |
|  | - | |
| Successful End Condition | Pengguna akan mendapat informasi tentang rapat | |
| Failed End Condition | Tidak muncul notifikasi | |
| Primary Actors | Pegawai | |
| Secondary Actors | - | |
| Trigger | Ketika sudah dapat notifikasi maka ketika di klik akan langsung menampilkan detail kegitan raat | |
| Main Flow | Step | Action |
|  | 1. | Notifikasi muncul |
|  | 2. | Klik tampil informasi rapat |
| Extension | Step | Branching Action |
|  |  | - |
|  |  |  |

## Model Activity Diagram

Berikut ini merupakan model activity diagram

* **Model Activity Diagram Login**

****

**Model Activity Diagram Lihat daftar kegiatan rapat**

****

## Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional dalam penelitian ini menggunakan pendekatan OOAD (*Object Oriented Analysis and Design*). Pemaparan lebih detail akan dipaparkan pada BAB 4.

# MODEL ANALISIS

## Diagram Class Analisis

Berikut ini adalah diagram kelas analisis yang terdapat pada Info Bappeda :

## Identifikasi Kelas Analisis

Berikut ini adalah daftar seluruh kelas-kelas analisis dalam tabel berikut:

Tabel 20 Kelas Analisis

|  |  |  |
| --- | --- | --- |
| Nama Kelas | Daftar Tanggung-Jawab (Method) | Daftar Atribut |
| LoginActivty | login() | Nip  Password |
| SplasScreen | isConnect() |  |
| MainActivity | onCreate()  onBackPresed()  onNavigationItemSelected()  loadFragment() |  |
| KegiatanAdapter | getKeigatanList()  onBindViewHolder()  getNamaKegiatan()  getWaktuMulai()  getWaktuSelesai()  getTanggal()  getTempat()  getAsalsurat() | Nama Kegiatan  Waktu mulai  Waltu selesai  Tanggal  Tempat  Asalsurat |
| KegiatanHomeAdapter | getKeigatanList()  onBindViewHolder()  getNamaKegiatan()  getWaktuMulai()  getWaktuSelesai()  getTanggal()  getTempat()  getAsalsurat() | Nama Kegiatan  Waktu mulai  Waltu selesai  Tanggal  Tempat  Asalsurat |
| DetailActivity | getTvJudulnya()  getTvPukulmulai()  getTvPukulSelesai()  getTvTanggalnya()  getTvTempatnya()  getTvAsalSurat() | Waktu mulai  Waltu selesai  Tanggal  Tempat  Asalsurat |
| ProfileFragment | txt\_nip()  txt\_nama()  txt\_email()  txt\_hp()  imgEdit() | NIP  Nama  Email  No Hp  Image |
| Kegiatanfragment | getData()  cickItemDetail()  reloadView() |  |
| HomeFragment | updateToken()  getData()  clickItemDetil()  reloadView() |  |
| GetKegiatan | GetResult() |  |
| User | getIdUser()  getNama()  getPassword()  getNip()  getEmail()  getNoHp() | Id User  Nama  Password  Nip  Email  NoHP |
| UserRespon | getData()  getMessage() |  |
| Kegiatan | getTvJudulnya()  getTvPukulmulai()  getTvPukulSelesai()  getTvTanggalnya()  getTvTempatnya()  getTvAsalSurat() | Waktu mulai  Waltu selesai  Tanggal  Tempat  Asalsurat |

## Realisasi Kelas Terhadap Use Case

Bagian ini diisi dengan bentuk realisasi hubungan kelas analisis terhadap Use Case

Berikut ini merupakan tabel realisasi kelas terhadap use case

Tabel 21 Realisasi Kelas Terhadap Use Case

|  |  |  |
| --- | --- | --- |
| No | Use Case | Realisasi Kelas |
| 1 | Login | Daftar kegiatan rapat, notifikasi kegiatan rapat |
| 2 | Daftar List kegiatan | Detail kegiatan rapat,Notifikasi kegiatan |
| 3 | Profile Pegawai | Edit profile, daftar kegiatan rapat |
| 4 | Notifikasi kegiatan | Daftar kegitan rapat, Detail kegiatan |

### Identifikasi Kelas Analisis tiap Paket

Berikut ini merupakan tabel identifikasi kelas analisis tiap paket

Tabel 22 Identifikasi Kelas Analisis tiap Paket

|  |  |  |  |
| --- | --- | --- | --- |
| No | Nama Paket | Nama Kelas Analisis | Jenis Kelas |
| 1 | adapter | KegiatanAdapter  KegiatanAdapterHome | Control Datas |
| 2 | model | Kegiatan  Login  User  UserRespon |  |
| 3 | rest | Api  ApiInterface  ApiUrl | Rest API |
| 4 | fragment | HomeFragment  KegiatanFragment  ProfileFragment | View |
| 5 | firebase | MyFirebaseInstanceIDServices  MyFirebaseMessagingServices | Service |
| 6 | activity | MainActivity  LoginActivity  DetailActivty  SplashScreen | Control |

# Model Perancangan

Berikut ini merupakan macam – macam model perancangan :

## Realisasi Use Case Tahap Perancangan

Bagian ini mendetailkan realisasi model perancangan terhadap Use Case dari perangkat lunak yang dibangun.

### Use Case Login User

Berikut merupakan realisasi usecae <<>> dalam identifikasi kelas diagram

#### Identifikasi Kelas Perancangan

Berikut adalah daftar kelas analaisis :

|  |  |  |
| --- | --- | --- |
| No | Nama Kelas Perancangan | Nama Kelas Analisis Terkait |
| 1 | Adapter | Login |
| 2 | User | Login |
| 3 | MainActivity | Login |
| 4 | LoginActivity | Login |
| 5 | HomeFragment | Login |
| 6 | Api | Login |
| 7 | ApiInterface | Login |
| 8 | ApiUrl | Login |

#### Sequence Diagram

**Sequence Diagram DatabaseHelper**

### Use Case tampilkan Daftar Kegiatan Rapat

Berikut merupakan realisasi usecae <<>> dalam identifikasi kelas diagram

#### Identifikasi Kelas Perancangan

|  |  |  |
| --- | --- | --- |
| No | Nama Kelas Perancangan | Nama Kelas Analisis Terkait |
| 1 | KegiatanAdapterHome | Daftar List Kegiatan |
| 2 | MainActivity | Daftar List Kegiatan |
| 3 | Kegiatan | Daftar List Kegiatan |
| 4 | Api | Daftar List Kegiatan |
| 5 | GetKegiatan | Daftar List Kegiatan |
| 6 | KegiatanFragment | Daftar List Kegiatan |
| 7 | ItemClickListener | Daftar List Kegiatan |
| 8 | SeesionManager | Daftar List Kegiatan |
| 9 | DetailActivity | Daftar List Kegiatan |

#### Sequence Diagram

### Use Case Tampilkan Profile Pegawai

Berikut merupakan realisasi usecae <<>> dalam identifikasi kelas diagram

#### Identifikasi Kelas Perancangan

|  |  |  |
| --- | --- | --- |
| No | Nama Kelas Perancangan | Nama Kelas Analisis Terkait |
| 1 | MainActivity | Profile |
| 2 | ProfileFragment | Profile |
| 3 | Api | Profile |
| 4 | ApiInterface | Profile |
| 5 | User |  |
| 6 | UserRespon |  |

#### Sequence Diagram

### Use Case Tampilkan Notifikasi rapat

Berikut merupakan realisasi usecae <<>> dalam identifikasi kelas diagram

#### Identifikasi Kelas Perancangan

|  |  |  |
| --- | --- | --- |
| No | Nama Kelas Perancangan | Nama Kelas Analisis Terkait |
| 1 | MainActivity | Notifikasi |
| 2 | DetailActiviy | Notifikasi |
| 3 | HomeFragment | Notifikasi |
| 4 | ApiInterface | Notifikasi |
| 5 | Api | Notifikasi |
| 6 | MyFirebaseInstanceIDServices | Notifikasi |
| 7 | MyFirebaseMessagingServices | Notifikasi |

#### Sequence Diagram

## Perancangan Detil Elemen Logical View

### Kelas KegiatanAdapter

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onCreateViewHolder | **p**ublic | Dipanggil ketiga activity akan diload |
| onBindViewHolder | Public | Memanggil data |
| getItemCount | public | Mendapatkan jumlah item |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| ListViewHolder | public | ArrayList |
| tvId | Package | string |
| tvTanggal | Package | String |

### Kelas KegiatanHomeAdapter

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onCreateViewHolder | **p**ublic | Dipanggil ketiga activity akan diload |
| onBindViewHolder | Public | Memanggil data |
| getItemCount | public | Mendapatkan jumlah item |
| onClick | public | Fungsi untuk klik detail |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| ListViewHolder | private | ArrayList |
| tvId | Package | string |
| tvTanggal | Package | String |

### Kelas GetKegiatan

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| getResult | public | Mendapatkan respon dari API |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
|  |  |  |

### Kelas SplashScreenActivity

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onCreate | Protected | Dipanggil saat menload SplashScreenActivity |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |
| Activity\_splash\_screen | public | LinearLayout |

### Kelas Kegiatan

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| Kegiatan | Public | Dipanggil saat get data |
| getIdKegiatan | Public | Dipanggil saat get id kegiatan |
| getNamaKegiatan | Public | Dipanggil saat get Nama kegiatan |
| getTempat | Public | Dipanggil saat get tempat |
| getAsalsurat | Public | Dipanggil saat get asal surat |
| getWaktuMulai | Public | Dipanggil saat get waktu mulai |
| getWaktuSelesai | Public | Dipanggil saat get waktu selesai |
| getTanggal | Public | Dipanggil saat get waktu tanggal |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |
| idKegiatan | public | String |
| namaKegiatan | public | String |
| tempat | public | String |
| asalsurat | public | String |
| waktuMulai | public | String |
| waktuSelesai | public | String |
| tanggal | public | String |

### Kelas Login

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| getResponse | Public | Mendapat response API |
| getDataUser | Public | Mendapat response API User |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |

### Kelas User

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| Kegiatan | Public | Dipanggil saat get data |
| getIdUser | Public | Dipanggil saat get id User |
| getNama | Public | Dipanggil saat get Nama |
| getPassword | Public | Dipanggil saat get Password |
| getNip | Public | Dipanggil saat get NIP |
| getNohp | Public | Dipanggil saat get No Hp |
| getEmail | Public | Dipanggil saat get Email |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |
| idUser | public | String |
| nama | public | String |
| password | public | String |
| nip | public | String |
| nohp | public | String |
| Email | public | String |

### Kelas UserRespon

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| getData | Public | Mendapat response API |
| getMessage | Public | Mendapat response API User |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |

### Kelas MainActivity

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onCreate | Protected | Dugunakan saat activity di panggil |
| onBackPressed | Public | Dipanggil saat back button |
| onNavigationItemSelected | Public | Dipanggil saat untuk bottom navigation |
| loadFragment | Public | Dipanggil untuk memanggil fragment |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |

### Kelas DetailActivity

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onCreate | protected | Dugunakan saat activity di panggil |
| getData | private | Dipanggil saat back button |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |
| getTvJudulnya | private | string |
| getTvPukulmulai | private | string |
| getTvPukulSelesai | private | string |
| getTvAsalSurat | private | string |
| getTvTanggalnya | private | string |
| getTvTempatnya | private | string |

### Kelas MyFirebaseInstanceIDServices

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onTokenRefresh | Public | Digunakan saat merefresh token device |
| storeToken | Private |  |
| sendRegistrationToServer | Public |  |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |

### Kelas MyFirebaseMessagingServices

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onMessageReceived | Public | Digunakan saat mendapatkan notif dari firebase |
| sendNotification | Private | Pengirim notif ke device |
| sendMessage | Private | Mengirim pesan ke device |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |
|  |  |  |

### Kelas HomeFragment

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onCreateView | protected | Dugunakan saat fragment di panggil |
| setImageForPosition | private |  |
| updateToken | private | Update token notif |
| onResponse | public | Menerima message sukses |
| onFailure | public | Menerima message gagal |
| getData | private | Menerima data dari API |
| clickItemDetail | private | Click detail |
| reloadView | public | Click detail |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |
| kegiatanList | private | ArrayList |
| sessionManager | private | static |
| rvKegiatan | private | static |

### Kelas KegiatanFragment

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| onCreateView | protected | Dugunakan saat fragment di panggil |
| onClick | private | Click Pindah activity |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |
| btnKeluar | private | ArrayList |
| txt\_nip | public | string |
| txt\_nama | public | string |
| txt\_email | public | string |
| txt\_hp | public | string |
| imgEdit | public | boolean |

### Kelas Api

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| getUrl | private | Digunakan untuk ket URL API |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |

### Kelas ApiUrl

|  |  |  |
| --- | --- | --- |
| **Nama Operasi** | **Visibility (private, public)** | **Keterangan** |
| getClient | private | Digunakan untuk memanggil library retrofit |
| **Nama Atribut** | **Visibility (private, public)** | **Tipe** |
| FEATURE\_NO\_TITLE | public | static |

## Algoritma / Query

Nama Kelas : MainActivity

Nama Operasi :

Algoritma :

package com.bappedajabar.infobapeddapp.activity;  
  
import android.animation.ObjectAnimator;  
import android.os.Build;  
import android.os.Bundle;  
  
import com.bappedajabar.infobapeddapp.R;  
import com.bappedajabar.infobapeddapp.fragment.HomeFragment;  
import com.bappedajabar.infobapeddapp.fragment.KegiatanFragment;  
import com.bappedajabar.infobapeddapp.fragment.ProfileFragment;  
import com.bappedajabar.infobapeddapp.rest.SessionManager;  
import com.google.android.material.bottomnavigation.BottomNavigationView;  
import com.google.firebase.messaging.FirebaseMessaging;  
//import com.google.firebase.messaging.FirebaseMessaging;  
  
import androidx.appcompat.app.AppCompatActivity;  
import androidx.annotation.NonNull;  
import androidx.core.content.ContextCompat;  
import androidx.fragment.app.Fragment;  
  
import android.view.MenuItem;  
import android.widget.TextView;  
import android.widget.Toast;  
  
public class MainActivity extends AppCompatActivity implements BottomNavigationView.OnNavigationItemSelectedListener {  
 private TextView mTextMessage;  
 private static final int *TIME\_INTERVAL* = 2000; // # milliseconds, desired time passed between two back presses.  
 private long mBackPressed;  
  
  
 SessionManager sessionManager;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*LOLLIPOP*) {  
 int startColor = getWindow().getStatusBarColor();  
 int endColor = ContextCompat.*getColor*(MainActivity.this, R.color.*colorPrimary*);  
 ObjectAnimator.*ofArgb*(getWindow(), "statusBarColor", startColor, endColor).start();  
 }  
  
 sessionManager = new SessionManager(MainActivity.this);  
  
 FirebaseMessaging.*getInstance*().subscribeToTopic("aca");  
  
  
 loadFragment(new HomeFragment());  
 BottomNavigationView bottomNavigationView = findViewById(R.id.*nav\_view*);  
 bottomNavigationView.setOnNavigationItemSelectedListener(this);  
 }  
  
  
  
  
  
 @Override  
 public void onBackPressed()  
 {  
 if (mBackPressed + *TIME\_INTERVAL* > System.*currentTimeMillis*())  
 {  
 super.onBackPressed();  
 return;  
 }  
 else { Toast.*makeText*(getBaseContext(), "Tap back button in order to exit", Toast.*LENGTH\_SHORT*).show(); }  
  
 mBackPressed = System.*currentTimeMillis*();  
 }  
  
  
  
 @Override  
 public boolean onNavigationItemSelected(@NonNull MenuItem menuItem) {  
 Fragment fragment = null;  
 switch (menuItem.getItemId()) {  
 case R.id.*navigation\_home*:  
 fragment = new HomeFragment();  
 break;  
 case R.id.*navigation\_kegiatan*:  
 fragment = new KegiatanFragment();  
 break;  
 case R.id.*navigation\_profile*:  
 fragment = new ProfileFragment();  
 break;  
 }  
 return loadFragment(fragment);  
 }  
  
  
// @Override  
// public boolean onCreateOptionsMenu(Menu menu) {  
// MenuInflater inflater = getMenuInflater();  
// inflater.inflate(R.menu.menu, menu);  
// return true;  
// }  
  
 private boolean loadFragment(Fragment fragment) {  
 if (fragment != null) {  
 getSupportFragmentManager().beginTransaction()  
 .replace(R.id.*fl\_container*, fragment)  
 .commit();  
 return true;  
 }  
 return false;  
 }  
  
  
}

Nama Kelas : DetailActivity

Nama Operasi : OnCreate dan sendNotification

Algoritma :

package com.bappedajabar.infobapeddapp.activity;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.os.Bundle;  
import android.widget.TextView;  
  
import com.bappedajabar.infobapeddapp.R;  
  
import butterknife.BindView;  
import butterknife.ButterKnife;  
  
public class DetailActivity extends AppCompatActivity {  
  
 @BindView(R.id.*tv\_judul\_detail*)  
 TextView tvJudulnya;  
 @BindView(R.id.*pukul\_mulai*)  
 TextView tvPukulmulai;  
 @BindView(R.id.*pukul\_selesai*)  
 TextView tvPukulSelesai;  
 @BindView(R.id.*tanggal\_detail*)  
 TextView tvTanggalnya;  
 @BindView(R.id.*tempat\_detail*)  
 TextView tvTempatnya;  
 @BindView(R.id.*asalsurat\_detail*)  
 TextView tvAsalSurat;  
  
 private TextView getTvJudulnya, getTvPukulmulai,getTvPukulSelesai,getTvTanggalnya,getTvTempatnya,getTvAsalSurat;  
 private String sJudul, sPukulMulai,sPukulSelesai,sTanggal,sTempat,sAsal;  
 String id;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_detail*);  
 id = getIntent().getStringExtra("id\_kegiatan");  
 ButterKnife.*bind*(this);  
  
 getTvJudulnya = findViewById(R.id.*tv\_judul\_detail*);  
 getTvPukulmulai = findViewById(R.id.*pukul\_mulai*);  
 getTvPukulSelesai = findViewById(R.id.*pukul\_selesai*);  
 getTvTanggalnya = findViewById(R.id.*tanggal\_detail*);  
 getTvTempatnya = findViewById(R.id.*tempat\_detail*);  
 getTvAsalSurat = findViewById(R.id.*asalsurat\_detail*);  
  
 getData();  
  
 }  
  
 private void getData(){  
  
 sJudul = getIntent().getStringExtra("nama");  
 sPukulMulai = getIntent().getStringExtra("waktu");  
 sPukulSelesai = getIntent().getStringExtra("waktus");  
 sTanggal = getIntent().getStringExtra("tanggal");  
 sTempat = getIntent().getStringExtra("tempat");  
 sAsal = getIntent().getStringExtra("asalsurat");  
  
 tvJudulnya.setText(sJudul);  
 tvPukulmulai.setText(sPukulMulai);  
 tvPukulSelesai.setText(sPukulSelesai);  
 tvTanggalnya.setText(sTanggal);  
 tvTempatnya.setText(sTempat);  
 tvAsalSurat.setText(sAsal);  
  
 }  
}

Nama Kelas : SpalshScreen

Nama Operasi :

Algoritma :

package com.bappedajabar.infobapeddapp.activity;  
  
import androidx.appcompat.app.AlertDialog;  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.content.Context;  
import android.content.DialogInterface;  
import android.content.Intent;  
import android.net.ConnectivityManager;  
import android.net.NetworkInfo;  
import android.os.Bundle;  
import android.os.Handler;  
import android.view.Window;  
import android.widget.Toast;  
  
import com.bappedajabar.infobapeddapp.R;  
import com.bappedajabar.infobapeddapp.rest.SessionManager;  
  
public class SplashScreen extends AppCompatActivity {  
 SessionManager sessionManager;  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 if (!isConnected(SplashScreen.this)) buildDialog(SplashScreen.this).show();  
 else {  
// Toast.makeText(SplashScreen.this,"Welcome", Toast.LENGTH\_SHORT).show();  
 setContentView(R.layout.*activity\_splash\_screen*);  
 sessionManager = new SessionManager(getApplicationContext());  
 sessionManager.getIdUSer();  
 Handler handler = new Handler();  
 handler.postDelayed(new Runnable() {  
 @Override  
 public void run() {  
 sessionManager.checkLogin();  
 finish();  
 }  
 }, 3000);  
 }  
 }  
  
 public boolean isConnected(Context context) {  
 ConnectivityManager cm = (ConnectivityManager) context.getSystemService(Context.*CONNECTIVITY\_SERVICE*);  
 NetworkInfo mNetInfo = cm.getActiveNetworkInfo();  
  
 if (mNetInfo != null && mNetInfo.isConnectedOrConnecting()) {  
 android.net.NetworkInfo wifi = cm.getNetworkInfo(ConnectivityManager.*TYPE\_WIFI*);  
 android.net.NetworkInfo mobile = cm.getNetworkInfo(ConnectivityManager.*TYPE\_MOBILE*);  
 if ((mobile != null && mobile.isConnectedOrConnecting()) || (wifi != null && wifi.isConnectedOrConnecting()))  
 return true;  
 else return false;  
 } else  
 return false;  
 }  
  
 public AlertDialog.Builder buildDialog(Context c) {  
  
 AlertDialog.Builder builder = new AlertDialog.Builder(c);  
 builder.setTitle("Tidak Ada Koneksi Internet");  
 builder.setMessage("Anda perlu cek kembali jaringan internet anda . Tekan ok untuk keluar");  
  
 builder.setPositiveButton("Ok", new DialogInterface.OnClickListener() {  
  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 finish();  
 }  
 });  
  
 return builder;  
 }  
 }

Nama Kelas : GetKegiatan

Nama Operasi : NoteColumns

Algoritma :

package com.bappedajabar.infobapeddapp.model;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
import java.util.List;  
  
public class GetKegiatan {  
 @SerializedName("result")  
 @Expose  
 private List<Kegiatan> result = null;  
  
 public List<Kegiatan> getResult() {  
 return result;  
 }  
  
 public void setResult(List<Kegiatan> result) {  
 this.result = result;  
 }  
}

Nama Kelas : Kegiatan

Nama Operasi :

Algoritma :

package com.bappedajabar.infobapeddapp.model;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class Kegiatan {  
 @SerializedName("id\_kegiatan")  
 @Expose  
 private String idKegiatan;  
  
 @SerializedName("nama")  
 @Expose  
 private String namaKegiatan;  
  
 @SerializedName("tempat")  
 @Expose  
 private String tempat;  
  
 @SerializedName("asalsurat")  
 @Expose  
 private String asalsurat;  
  
 @SerializedName("waktu\_mulai")  
 @Expose  
 private String waktuMulai;  
  
 @SerializedName("waktu\_selesai")  
 @Expose  
 private String waktuSelesai;  
  
 @SerializedName("tanggal")  
 @Expose  
 private String tanggal;  
  
  
 public Kegiatan(){  
  
 }  
  
  
 public Kegiatan(String idKegiatan, String namaKegiatan, String tempat, String asalsurat, String waktuMulai, String waktuSelesai, String tanggal) {  
 this.idKegiatan = idKegiatan;  
 this.namaKegiatan = namaKegiatan;  
 this.tempat = tempat;  
 this.asalsurat = asalsurat;  
 this.waktuMulai = waktuMulai;  
 this.waktuSelesai = waktuSelesai;  
 this.tanggal = tanggal;  
  
 }  
  
 public String getIdKegiatan() {  
 return idKegiatan;  
 }  
  
 public void setIdKegiatan(String idKegiatan) {  
 this.idKegiatan = idKegiatan;  
 }  
  
 public String getNamaKegiatan() {  
 return namaKegiatan;  
 }  
  
 public void setNamaKegiatan(String namaKegiatan) {  
 this.namaKegiatan = namaKegiatan;  
 }  
  
 public String getTempat() {  
 return tempat;  
 }  
  
 public void setTempat(String tempat) {  
 this.tempat = tempat;  
 }  
  
 public String getAsalsurat() {  
 return asalsurat;  
 }  
  
 public void setAsalsurat(String asalsurat) {  
 this.asalsurat = asalsurat;  
 }  
  
 public String getWaktuMulai() {  
 return waktuMulai;  
 }  
  
 public void setWaktuMulai(String waktuMulai) {  
 this.waktuMulai = waktuMulai;  
 }  
  
 public String getWaktuSelesai() {  
 return waktuSelesai;  
 }  
  
 public void setWaktuSelesai(String waktuSelesai) {  
 this.waktuSelesai = waktuSelesai;  
 }  
  
 public String getTanggal() {  
 return tanggal;  
 }  
  
 public void setTanggal(String tanggal) {  
 this.tanggal = tanggal;  
 }  
  
  
}

Nama Kelas : Login

Nama Operasi :

Algoritma :

package com.bappedajabar.infobapeddapp.model;  
  
import com.google.gson.annotations.SerializedName;  
  
public class Login {  
 @SerializedName("message")  
 private String response;  
  
 //get data tanpa index array  
 @SerializedName("data")  
 User DataUser;  
  
 public String getResponse() {  
 return response;  
 }  
  
 public User getDataUser() {  
 return DataUser;  
 }  
}

Nama Kelas : User

Nama Operasi : update

Algoritma :

package com.bappedajabar.infobapeddapp.model;  
  
import com.google.gson.annotations.Expose;  
import com.google.gson.annotations.SerializedName;  
  
public class User {  
 @SerializedName("id\_user")  
 @Expose  
 private int idUser;  
  
 @SerializedName("nip")  
 @Expose  
 private String nip;  
  
 @SerializedName("nama")  
 @Expose  
 private String nama;  
 @SerializedName("password")  
 @Expose  
 private String password;  
 @SerializedName("no\_hp")  
 @Expose  
 private String nohp;  
 @SerializedName("email")  
 @Expose  
 private String email;  
 @SerializedName("status")  
 @Expose  
 private String status;  
 @SerializedName("akses")  
 @Expose  
 private String akses;  
  
 @SerializedName("data")  
 private Boolean error;  
  
 @SerializedName("message")  
 private String message;  
  
 @SerializedName("user")  
 private User user;  
  
 public User(Boolean error, String message, User user) {  
 this.error = error;  
 this.message = message;  
 this.user = user;  
 }  
  
 public User(String name, String nip, String email, String password, String hp) {  
 this.nama = name;  
 this.nip = nip;  
 this.email = email;  
 this.password = password;  
 this.nohp = hp;  
  
 }  
  
 public Boolean getError() {  
 return error;  
 }  
  
 public String getMessage() {  
 return message;  
 }  
  
 public User getUser() {  
 return user;  
 }  
  
  
 public User(int idUser, String nama, String nip, String password,String nohp,String email,String status,String akses) {  
 this.idUser = idUser;  
 this.nama = nama;  
 this.nip = nip;  
 this.password = password;  
 this.nohp = nohp;  
 this.email = email;  
 this.status = status;  
 this.akses = akses;  
 }  
  
 public String getAkses() {  
 return akses;  
 }  
  
 public void setAkses(String akses) {  
 this.akses = akses;  
 }  
  
 public int getIdUser() {  
 return idUser;  
 }  
  
 public void setIdUser(int idUser) {  
 this.idUser = idUser;  
 }  
  
 public String getNama() {  
 return nama;  
 }  
  
 public void setNama(String nama) {  
 this.nama = nama;  
 }  
  
 public String getPassword() {  
 return password;  
 }  
  
 public void setPassword(String password) {  
 this.password = password;  
 }  
  
 public String getNip() {  
 return nip;  
 }  
  
 public void setNip(String nip) {  
 this.nip = nip;  
 }  
  
 public String getNohp() {  
 return nohp;  
 }  
  
 public void setNohp(String nohp) {  
 this.nohp = nohp;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getStatus() {  
 return status;  
 }  
  
 public void setStatus(String status) {  
 this.status = status;  
 }  
}

Nama Kelas : UserRespon

Nama Operasi :

Algoritma :

package com.bappedajabar.infobapeddapp.model;  
  
//import javax.annotation.Generated;  
import com.google.gson.annotations.SerializedName;  
  
//@Generated("net.hexar.json2pojo")  
@SuppressWarnings("unused")  
public class UserRespon {  
  
 @SerializedName("data")  
 private User mData;  
 @SerializedName("message")  
 private String mMessage;  
  
 public User getData() {  
 return mData;  
 }  
  
 public void setData(User data) {  
 mData = data;  
 }  
  
 public String getMessage() {  
 return mMessage;  
 }  
  
 public void setMessage(String message) {  
 mMessage = message;  
 }  
  
}

Nama Kelas : MyFirebaseInstanceIDServices

Nama Operasi :

Algoritma :

package com.bappedajabar.infobapeddapp.fierbase;  
  
import android.util.Log;  
  
  
import com.google.firebase.iid.FirebaseInstanceId;  
import com.google.firebase.iid.FirebaseInstanceIdService;  
  
public class MyFirebaseInstanceIDServices extends FirebaseInstanceIdService {  
 private static final String *TAG* = "MyFirebaseIIDService";  
  
 @Override  
 public void onTokenRefresh() {  
 String refreshedToken = FirebaseInstanceId.*getInstance*().getToken();  
 Log.*d*(*TAG*,"Refreshed Token : "+refreshedToken);  
  
  
 sendRegistrationToServer(refreshedToken);  
  
 //calling the method store token and passing token  
 storeToken(refreshedToken);  
 }  
 private void storeToken(String token) {  
 //we will save the token in sharedpreferences later  
// SharedPrefManager.getInstance(getApplicationContext()).saveDeviceToken(token);  
 }  
  
  
  
 private void sendRegistrationToServer(String refreshedToken) {  
 // *TODO: Implement this method to send token to your app server.* }  
}

Nama Kelas : MyFirebaseMessagingServices

Nama Operasi :

Algoritma :

package com.bappedajabar.infobapeddapp.fierbase;  
  
import android.app.NotificationChannel;  
import android.app.NotificationManager;  
import android.app.PendingIntent;  
import android.content.Context;  
import android.content.Intent;  
import android.graphics.Color;  
import android.media.RingtoneManager;  
import android.net.Uri;  
import android.util.Log;  
  
import androidx.core.app.NotificationCompat;  
  
import com.bappedajabar.infobapeddapp.R;  
import com.bappedajabar.infobapeddapp.activity.DetailActivity;  
import com.bappedajabar.infobapeddapp.activity.MainActivity;  
import com.bappedajabar.infobapeddapp.rest.SessionManager;  
import com.google.firebase.messaging.FirebaseMessagingService;  
import com.google.firebase.messaging.RemoteMessage;  
  
public class MyFirebaseMessagingServices extends FirebaseMessagingService {  
 private static final String *TAG* = "MyFirebaseMsgService";  
 SessionManager sessionManager;  
 Context context;  
 @Override  
 public void onMessageReceived(RemoteMessage remoteMessage) {  
  
  
 Log.*d*(*TAG*, "REMOTEDATA: " + remoteMessage.getData().size());  
  
 if(remoteMessage.getData().size()>0){  
 Log.*d*(*TAG*, "Message data payload: " + remoteMessage.getData().size());  
  
 String activity = remoteMessage.getData().get("click\_action");  
 String body = remoteMessage.getData().get("body");  
 sendNotification(body,activity , remoteMessage);  
 }  
  
 // Check if message contains a notification payload.  
 if (remoteMessage.getNotification() != null) {  
 Log.*d*(*TAG*, "Message Notification Body: " + remoteMessage.getNotification().getBody());  
  
 }  
 }  
  
 private void sendNotification(String body , String activity , RemoteMessage remoteMessage ) {  
 NotificationManager notificationManager =  
 (NotificationManager) getSystemService(Context.*NOTIFICATION\_SERVICE*);  
 Intent intent = new Intent();  
 Intent intentNew = sendMessage(activity,intent,remoteMessage);  
 PendingIntent pendingIntent = PendingIntent.*getActivity*(this, 0, intentNew,  
 PendingIntent.*FLAG\_ONE\_SHOT*);  
 Uri defaultSoundUri= RingtoneManager.*getDefaultUri*(RingtoneManager.*TYPE\_NOTIFICATION*);  
 NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder(this)  
 .setSmallIcon(R.mipmap.*ic\_launcher*)  
 .setContentTitle("Info Bappeda App")  
 .setContentText(body)  
 .setAutoCancel(true)  
 .setSound(defaultSoundUri)  
 .setContentIntent(pendingIntent);  
  
 if (android.os.Build.VERSION.*SDK\_INT* >= android.os.Build.VERSION\_CODES.*O*) {  
 int importance = NotificationManager.*IMPORTANCE\_HIGH*;  
 NotificationChannel notificationChannel = new NotificationChannel(body, "NOTIFICATION\_CHANNEL\_NAME", importance);  
 notificationChannel.enableLights(true);  
 notificationChannel.setLightColor(Color.*GREEN*);  
 notificationChannel.enableVibration(true);  
 notificationChannel.setVibrationPattern(new long[] {100, 200, 300, 400, 500, 400, 300, 200, 400});  
 notificationBuilder.setChannelId(body);  
 notificationManager.createNotificationChannel(notificationChannel);  
 }  
  
  
 notificationManager.notify(1, notificationBuilder.build());  
 }  
  
  
 private Intent sendMessage(String activity, Intent intent , RemoteMessage remoteMessage){  
 if(activity.equals("KEGIATANACTIVITY")){  
 intent = new Intent(this, DetailActivity.class);  
 intent.putExtra("id\_kegiatan",remoteMessage.getData().get("id\_kegiatan"));  
 intent.addFlags(Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP*);  
 }  
 return intent;  
 }  
}