BAB III

METODOLOGI DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Pada penelitian ini terdapat beberapa tahap yang dilakukan untuk mencapai tujuan penelitian. Tahap-tahapnya adalah sebagai berikut.

1) Studi Literatur

Penelitian diawali dengan mencari dan mempelajari buku, jurnal, artikel, *e-book*, dan halaman web. Tahap ini menekankan pada pematangan konsep-konsep yang dibutuhkan dalam penelitian, seperti *android notification service*, *push technology*, Google Cloud Messaging, *pull technology*, GCM Network Manager, dan kriteria dan metode pengukuran notifikasi *mobile*.

2) Implementasi

Tahap yang dilakukan setelah mematangkan konsep untuk penelitian adalah implementasi. Pada tahap ini dilakukan perancangan sistem terlebih dahulu sebelum melakukan *coding* pembuatan sistem. Setelah sistem selesai dibuat, aplikasi diuji coba pada beberapa perangkat Android berbeda. Tujuan dari tahap ini adalah untuk menemukan *bug* pada sistem agar dapat diperbaiki sebelum pengumpulan data dilakukan.

3) Uji Coba Aplikasi

Pada tahap ini akan dilakukan metode pengukuran dengan *field studies* bersamaan dengan *hands-on measurement* untuk mendapatkan *click ratio* dan *click time* dari notifikasi-notifikasi yang didapatkan oleh responden penelitian. Uji coba aplikasi akan dilakukan selama dua minggu untuk setiap responden, dengan satu minggu untuk mendapatkan data-data untuk satu metode (Kuniavsky, 2007).

4) Analisa Data

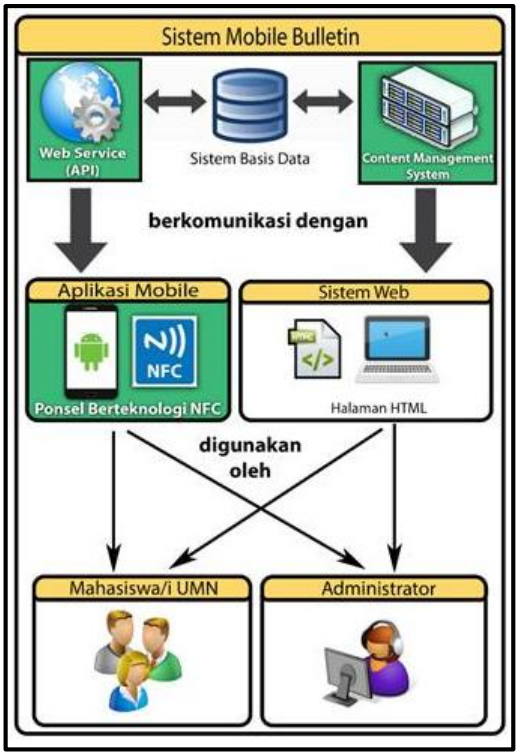
Pada tahap ini, data penelitian yang berupa jawaban kuisioner, *click ratio*, dan *click time* akan dianalisa menjadi bentuk kuantitatif agar dapat dibandingkan nilai untuk metode *push* dan *pull* yang didapat pada tahap uji coba.

5) Penulisan Laporan

Merupakan tahap akhir dalam penelitian, yaitu menuangkan setiap kegiatan yang dilakukan dalam penelitian dalam bentuk tulisan yang berupa laporan skripsi.

3.2 Perancangan Sistem

Pengembangan Sistem UMN Bulletin dengan fitur notifikasi meliputi pengembangan Aplikasi UMN Bulletin, Content Management System (CMS) UMN Bulletin, dan Application Program Interface (API) UMN Bulletin. Gambar 3.1 merupakan arsitektur sistem yang dikembangkan pada penelitian ini.



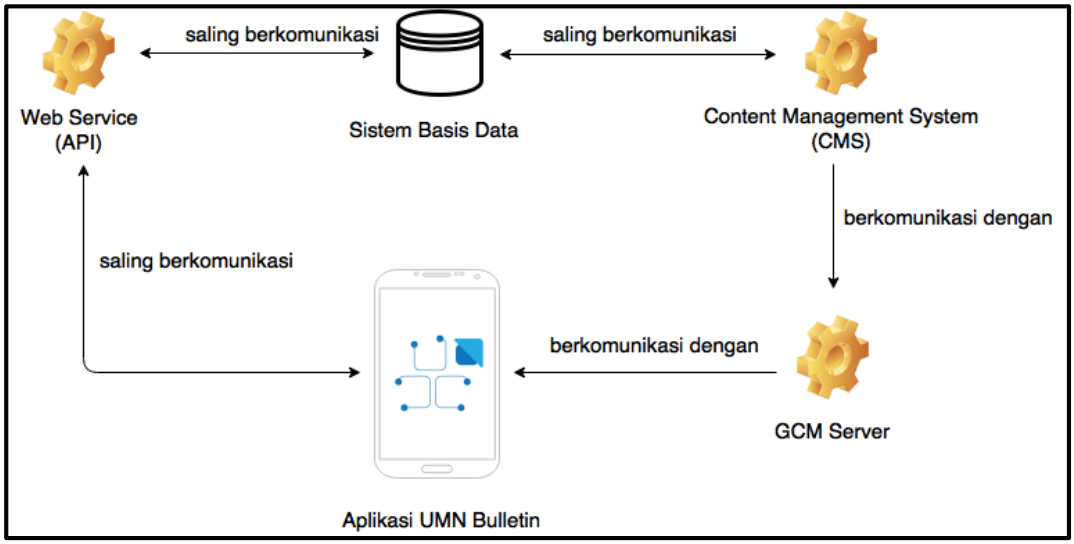
Gambar 3.1 Arsitektur Sistem
(Sumber: Audy, 2016 : 39)

Pengembangan aplikasi UMN Bulletin meliputi beberapa penambahan *class* yang digunakan untuk membantu dalam implementasi metode *push* dan *pull*, yaitu RegistrationClientService, RegistrationClient, dan SettingsFragment. *Class* yang secara khusus digunakan untuk mengimplementasikan metode *push* terdiri dari RegistrationIntentService, MyGcmListenerService, MyInstanceIDListenerService, dan RegistrationGcm, sedangkan *class* yang secara khusus digunakan untuk mengimplementasikan metode *pull* terdiri dari RegistrationPullClientService, PullService, MyGnmTaskService, RegistrationPullClient, PullResponse, dan PullNotification.

Pengembangan CMS UMN Bulletin meliputi beberapa penambahan *class* yang berfungsi sebagai *model*, yaitu Client, GcmClient, PullClient, NotificationPush, dan NotificationPull. *Class* yang berfungsi sebagai *model* ini juga ditambahkan pada pengembangan API UMN Bulletin. Selain *model*, pada API UMN Bulletin juga terdapat penambahan *class* yang berfungsi sebagai *controller*, yaitu NotificationPushController dan NotificationPullController. Penjelasan mengenai *class-class* ini dapat dilihat pada bagian Class Diagram untuk masing-masing sistem.

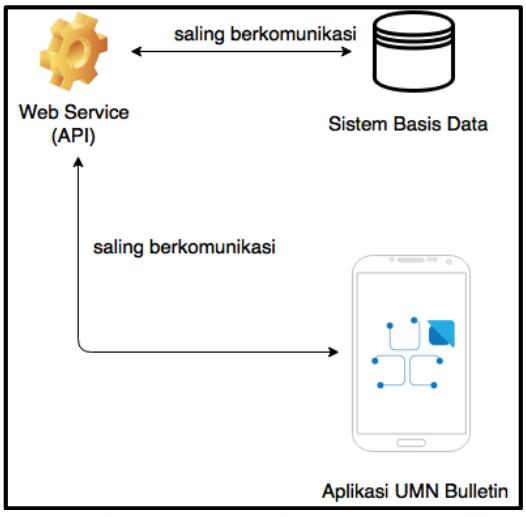
Gambar 3.2 merupakan arsitektur sistem yang digunakan untuk metode *push*. Terdapat API dan CMS yang berkomunikasi dengan Sistem Basis Data, CMS yang berkomunikasi dengan GCM Server untuk mengirimkan data notifikasi, GCM Server yang berkomunikasi dengan Aplikasi UMN Bulletin untuk meneruskan data notifikasi dari CMS, dan Aplikasi UMN Bulletin yang saling berkomunikasi dengan API untuk mendaftarkan pengguna dan hal-hal lain yang berhubungan dengan notifikasi untuk metode *push*, seperti meng-*update* waktu data notifikasi

diterima dan meng-*update* waktu notifikasi diklik.



Gambar 3.2 Arsitektur Sistem Push

Gambar 3.3 merupakan arsitektur sistem yang dibutuhkan untuk metode *pull*. Terdapat tiga komponen untuk arsitektur sistem ini, yaitu Sistem Basis Data, API, dan Aplikasi UMN Bulletin. API saling berkomunikasi dengan Sistem Basis Data dan Aplikasi UMN Bulletin saling berkomunikasi dengan API untuk mendaftarkan pengguna dan hal-hal yang berhubungan dengan data notifikasi untuk metode *pull*.



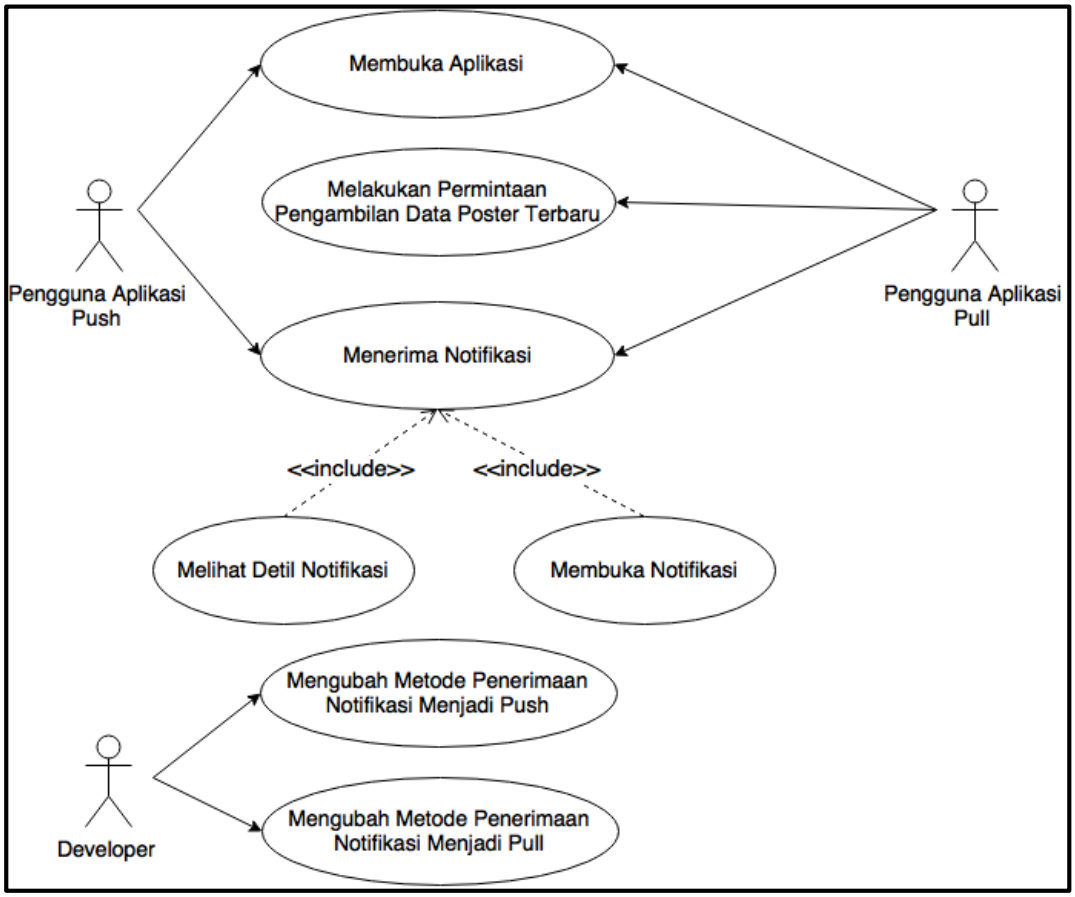
Gambar 3.3 Arsitektur Sistem Pull

Pada penelitian ini, pengembangan Aplikasi, CMS, dan API UMN Bulletin menggunakan pendekatan pemrograman berorientasi objek, sehingga akan

digunakan *Unified Modeling Language* (UML) untuk ketiga sistem tersebut. Selain itu, *Entity Relationship Diagram* (ERD) sebagai perancangan untuk *database* yang digunakan ketiga sistem tersebut dan desain antarmuka untuk notifikasi juga akan dijelaskan.

3.2.1 Perancangan Aplikasi UMN Bulletin

Tahap awal dalam pengembangan aplikasi UMN Bulletin adalah pembuatan *Use Case Diagram*. Gambar 3.4 merupakan *use case diagram* aplikasi UMN Bulletin dengan notifikasi.



Gambar 3.4 Use Case Diagram Aplikasi UMN Bulletin

Use Case Membuka Aplikasi dijelaskan pada Tabel 3.1.

Tabel 3.1 Use Case Description Membuka Aplikasi

Use Case Name	Membuka Aplikasi
Actor	Pengguna Aplikasi Push dan Pengguna Aplikasi Pull
Description	Merupakan sebuah *event* ketika aktor membuka aplikasi UMN Bulletin
Trigger	Aktor ingin membuka aplikasi UMN Bulletin
Normal Flow of Event	1. Aktor membuka aplikasi UMN Bulletin pada perangkat Androidnya. 2. Aplikasi akan mendaftarkan ID perangkat pengguna sebagai pengguna aplikasi ke server UMN Bulletin.
Pre Condition	-
Post Condition	Aplikasi pada perangkat aktor terdaftar sebagai pengguna aplikasi

Use Case Melakukan Permintaan Pengambilan Data Poster Terbaru dijelaskan pada Tabel 3.2.

Tabel 3.2 Use Case Description Melakukan Permintaan Pengambilan Data Poster Terbaru

Use Case Name	Melakukan Permintaan Pengambilan Data Poster Terbaru
Actor	Pengguna Aplikasi Pull
Description	Merupakan sebuah *event* dimana aplikasi akan melakukan permintaan pengambilan data poster terbaru ke server UMN Bulletin
Trigger	Waktu yang telah dijadwalkan tiba dan aktor mengaktifkan koneksi internet pada waktu tersebut (apabila belum aktif)
Normal Flow of Event	1. Aktor mengaktifkan koneksi internet pada waktu yang telah dijadwalkan (apabila belum aktif). 2. Apabila sudah aktif, aplikasi akan melakukan permintaan pengambilan data poster terbaru ke server UMN Bulletin.
Pre Condition	Developer mengubah metode penerimaan notifikasi menjadi *pull*
Post Condition	Aplikasi mendapatkan data notifikasi

Use Case Menerima Notifikasi dijelaskan pada Tabel 3.3.

Tabel 3.3 Use Case Description Menerima Notifikasi

Use Case Name	Menerima Notifikasi
Actor	Pengguna Aplikasi Push dan Pengguna Aplikasi Pull
Description	Merupakan sebuah *event* dimana aplikasi membuat sebuah notifikasi berdasarkan data notifikasi yang telah didapat
Trigger	Aplikasi berhasil mendapatkan data notifikasi dari server GCM (*push*) atau server UMN Bulletin (*pull*)
Normal Flow of Event	1. Aplikasi membuat notifikasi berdasarkan data notifikasi yang telah didapat, termasuk dengan detil informasinya. 2. Aplikasi menambahkan data notifikasi yang didapat ke database perangkat. 3. Aplikasi mengirimkan permintaan untuk meng-*update* waktu data notifikasi diterima ke server UMN Bulletin.
Pre Condition	Developer mengubah metode penerimaan notifikasi menjadi *push* (*push*) atau telah melakukan permintaan pengambilan data poster terbaru (*pull*)
Post Condition	Aktor mendapatkan notifikasi dari aplikasi UMN Bulletin

Use Case Melihat Detil Notifikasi dijelaskan pada Tabel 3.4.

Tabel 3.4 Use Case Description Melihat Detil Notifikasi

Use Case Name	Melihat Detil Notifikasi
Actor	Pengguna Aplikasi Push dan Pengguna Aplikasi Pull
Description	Merupakan sebuah *event* dimana aktor melihat detil informasi dari notifikasi yang telah didapat dari aplikasi UMN Bulletin
Trigger	Aktor ingin melihat detil informasi dari notifikasi
Normal Flow of Event	1. Aktor melakukan gerakan menyapu (*swipe*) notifikasi ke bawah dengan dua jari. 2. Sistem Android akan menampilkan detil informasi notifikasi seperti yang sudah di-*set* oleh aplikasi ketika mengirim notifikasi ke aktor.
Pre Condition	Aktor telah menerima notifikasi dari aplikasi UMN Bulletin
Post Condition	Aktor berhasil melihat detil informasi notifikasi

Use Case Membuka Notifikasi dijelaskan pada Tabel 3.5.

Tabel 3.5 Use Case Description Membuka Notifikasi

Use Case Name	Membuka Notifikasi
Actor	Pengguna Aplikasi Push dan Pengguna Aplikasi Pull
Description	Merupakan sebuah *event* dimana aktor membuka notifikasi yang telah didapat dari aplikasi UMN Bulletin
Trigger	Aktor ingin melihat detil informasi poster
Normal Flow of Event	1. Aktor mengklik notifikasi yang telah didapat. 2. Sistem Android akan membuka aplikasi UMN Bulletin pada laman detil poster. 3. Aplikasi UMN Bulletin akan mengirimkan waktu notifikasi diklik ke server UMN Bulletin.
Pre Condition	Aktor telah menerima notifikasi dari aplikasi UMN Bulletin
Post Condition	Aktor berhasil melihat detil informasi poster pada aplikasi UMN Bulletin

Use Case Mengubah Metode Penerimaan Notifikasi Menjadi Push dijelaskan pada Tabel 3.6.

Tabel 3.6 Use Case Description Mengubah Metode Penerimaan Notifikasi Menjadi Push

Use Case Name	Mengubah Metode Penerimaan Notifikasi Menjadi Push
Actor	Developer
Description	Merupakan sebuah *event* dimana aktor mengubah metode penerimaan notifikasi menjadi *push* pada perangkat pengguna aplikasi UMN Bulletin
Trigger	Developer ingin mengubah metode penerimaan notifikasi menjadi *push*
Normal Flow of Event	1. Aktor mengaktifkan Developer Mode pada laman Settings. 2. Aktor memasukkan password untuk Developer Mode. 3. Pilihan metode awal (*default*) adalah *pull*, sehingga aktor mengubah pilihan metode menjadi *push*.

Tabel 3.6 Use Case Description Mengubah Metode Penerimaan Notifikasi Menjadi Push (Lanjutan)

Normal Flow of Event	4. Aktor memasukkan password untuk mengubah metode. 5. Aktor menekan tombol Save. 6. Aplikasi akan mendaftarkan ID perangkat pengguna beserta dengan token registrasi GCM sebagai pengguna aplikasi *push* ke server UMN Bulletin.
Pre Condition	ID perangkat pengguna telah terdaftar sebagai pengguna aplikasi
Post Condition	Aktor berhasil mengubah metode penerimaan notifikasi menjadi *push*, sehingga pengguna aplikasi sudah dapat menerima notifikasi dengan metode *push*

Use Case Mengubah Metode Penerimaan Notifikasi Menjadi Pull dijelaskan pada Tabel 3.7.

Tabel 3.7 Use Case Description Mengubah Metode Penerimaan Notifikasi Menjadi Pull

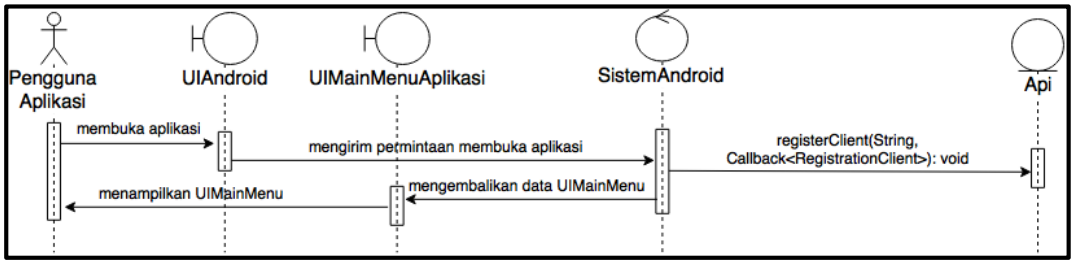
Use Case Name	Mengubah Metode Penerimaan Notifikasi Menjadi Pull
Actor	Developer
Description	Merupakan sebuah *event* dimana aktor mengubah metode penerimaan notifikasi menjadi *pull* pada perangkat pengguna aplikasi UMN Bulletin
Trigger	Developer ingin mengubah metode penerimaan notifikasi menjadi *pull*
Normal Flow of Event	1. Aktor mengaktifkan Developer Mode pada laman Settings. 2. Aktor memasukkan password untuk Developer Mode. 3. Pilihan metode awal (*default*) adalah *pull*, sehingga aktor langsung menekan tombol Save. 4. Aplikasi akan mendaftarkan ID perangkat pengguna sebagai pengguna aplikasi *pull* ke server UMN Bulletin. 5. Aplikasi akan menjadwalkan permintaan pengambilan data poster terbaru.

Tabel 3.7 Use Case Description Mengubah Metode Penerimaan Notifikasi Menjadi Pull (Lanjutan)

Pre Condition	ID perangkat pengguna telah terdaftar sebagai pengguna aplikasi
Post Condition	Aktor berhasil mengubah metode penerimaan notifikasi menjadi *pull*, sehingga pengguna aplikasi sudah dapat menerima notifikasi dengan metode *pull*

Dari *normal flow of event* yang ada pada tiap tabel *use case description*, dapat dibuat *Sequence Diagram*. Gambar 3.5 merupakan Sequence Diagram Membuka Aplikasi. Skenario untuk *event* tersebut:

1) Pengguna Aplikasi (*push* dan *pull*) membuka aplikasi Mobile Bulletin.
2) UserInterfaceAndroid (UI) akan mengirim permintaan pengguna ke Sistem Android.
3) Sistem Android mendaftarkan ID perangkat pengguna sebagai pengguna aplikasi melalui *class* Api dan mengembalikan UI berupa MainMenu dari aplikasi.
4) UIMainMenu menampilkan laman ke Pengguna Aplikasi.

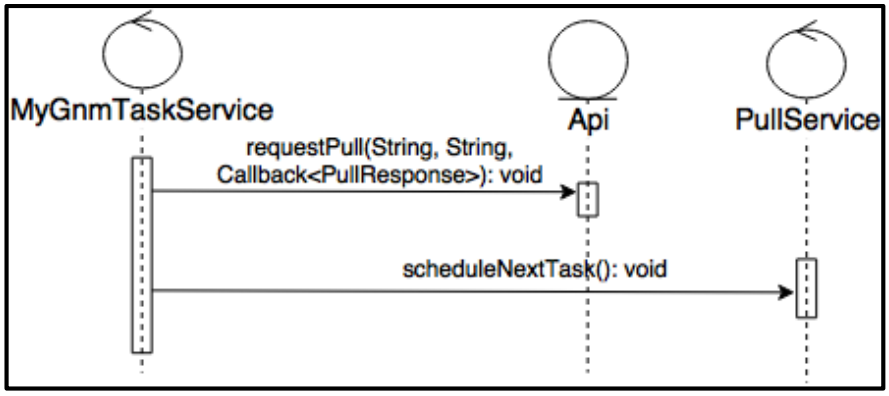


Gambar 3.5 Sequence Diagram Membuka Aplikasi

Gambar 3.6 merupakan Sequence Diagram Melakukan Permintaan Pengambilan Poster Baru. Skenario untuk *event* tersebut:

1) MyGnmTaskService akan memanggil fungsi requestPull(String, String, Callback<PullResponse>) pada *class* Api untuk mendapatkan data informasi poster terbaru.

2) Setelah itu, MyGnmTaskService akan memanggil fungsi scheduleNextTask() pada PullService untuk menjadwalkan *request* selanjutnya.

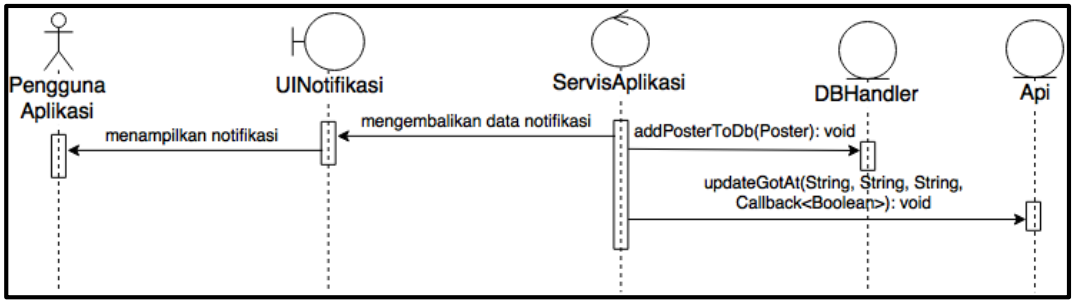


Gambar 3.6 Sequence Diagram Melakukan Permintaan Pengambilan Poster Baru

Gambar 3.7 merupakan merupakan Sequence Diagram Menerima Notifkasi. ServisAplikasi yang dimaksud adalah MyGcmListenerService untuk aplikasi *push* dan MyGnmTaskService untuk aplikasi *pull*. Skenario untuk *event* tersebut:

1) Setelah mendapatkan data notifikasi, ServisAplikasi akan mengirimkan data notifikasi ke UI untuk ditampilkan ke Pengguna Aplikasi.

2) Kemudian ServisAplikasi akan menambahkan data poster berdasarkan notifikasi ke *database* aplikasi dan meng-*update* waktu data notifikasi diterima sesuai metode ke *server* dengan bantuan *class* Api.



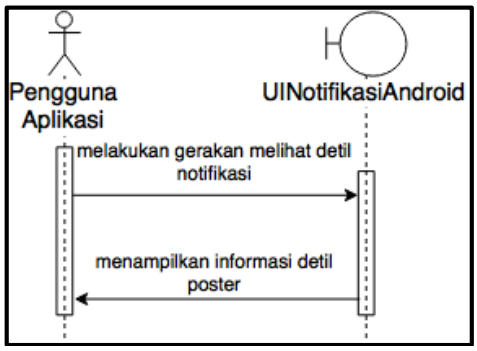
Gambar 3.7 Sequence Diagram Menerima Notifikasi

Gambar 3.8 merupakan Sequence Diagram Melihat Detil Notifikasi. Skenario untuk *event* tersebut:

1) Aktor yang telah mendapatkan notifikasi melakukan gerakan melihat detil

notifikasi dengan cara menyapu (*swipe*) notifikasi ke bawah dengan dua jari.

2) UI yang telah memiliki informasi detil poster langsung menampilkannya ke aktor, yaitu berupa nama dan gambar poster.



Gambar 3.8 Sequence Diagram Melihat Detil Notifikasi

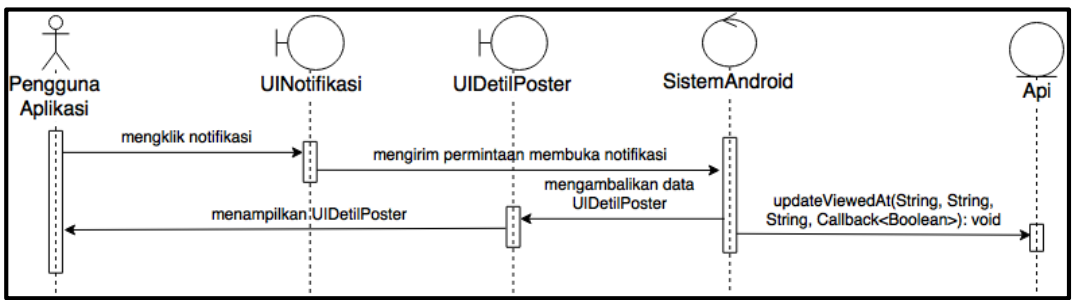
Gambar 3.9 merupakan Sequence Diagram Membuka Notifikasi. Skenario untuk *event* tersebut:

1) Aktor yang telah mendapat notifikasi, mengklik notifikasi pada UI.

2) UI meneruskan permintaan aktor ke SistemAndroid.

3) SistemAndroid mengembalikan data berupa UI laman detil poster untuk ditampilkan ke aktor.

4) Selanjutnya SistemAndroid meng-*update* data notifikasi dibuka dengan memanggil fungsi updateViewedAtPush(String, String, String, Callback<Boolean>) untuk aplikasi *push* dan updateViewedAtPull(String, String, String, Callback<Boolean>) untuk aplikasi *pull*.

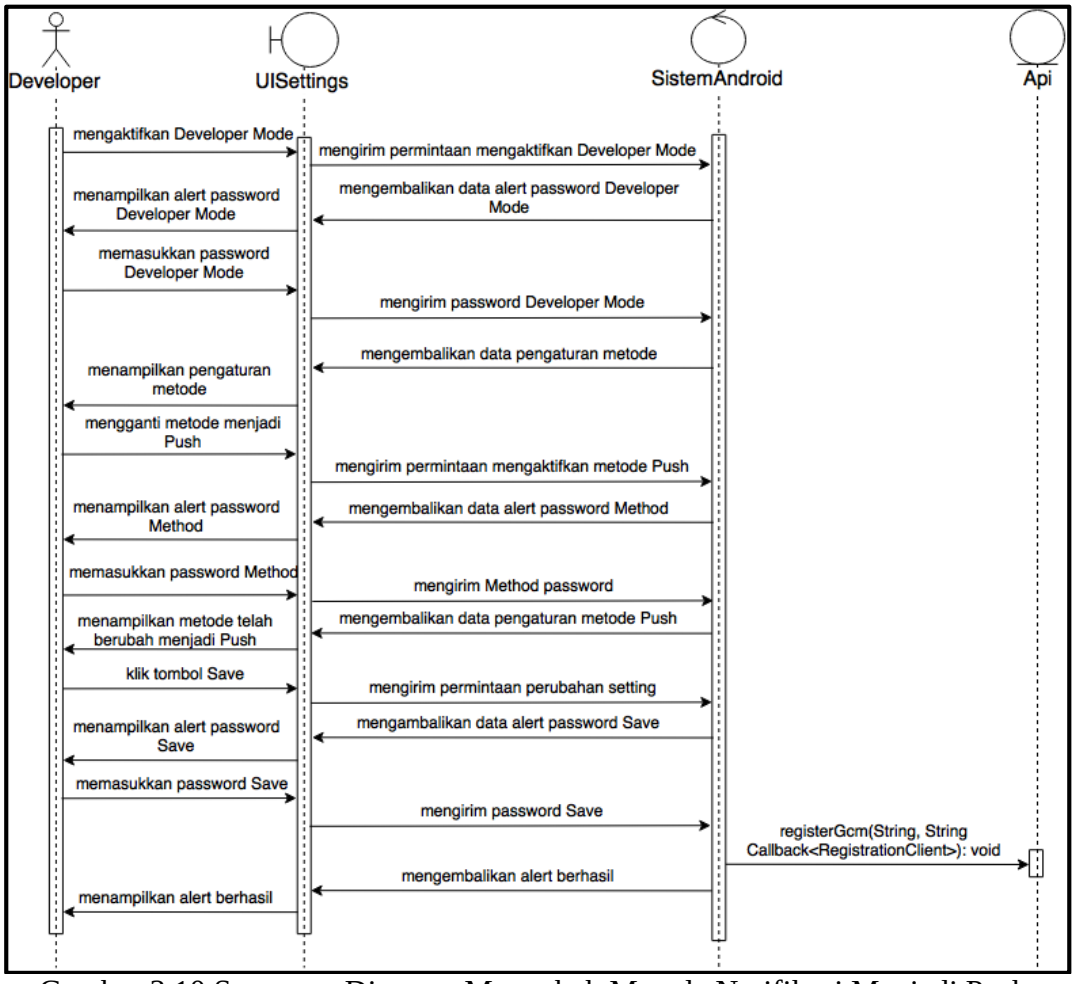


Gambar 3.9 Sequence Diagram Membuka Notifikasi

Gambar 3.10 merupakan Sequence Diagram Mengubah Metode Notifikasi

Menjadi Push. Skenario untuk *event* tersebut:

1) Developer yang berada pada UISettings, mengubah pilihan Developer Mode.
2) Selanjutnya Developer akan ditampilkan *alert* oleh UI untuk memasukkan *password* Developer Mode.
3) Kemudian UI akan menampilkan pilihan Method, dengan *default* Method adalah Pull.
4) Developer mengubah pilihan Method menjadi Push.
5) UI menampilkan *alert* untuk memasukkan *password* Method.
6) Developer memasukkan *password* Method, kemudian UI akan menampilkan perubahan pilihan Method menjadi Push.
7) Developer menekan tombol Save.
8) UI menampilkan *alert* untuk memasukkan *password* Save.
9) Developer memasukkan *password* Save.
10) Sistem Android akan mendaftarkan ID perangkat pengguna sebagai pengguna aplikasi *push*.

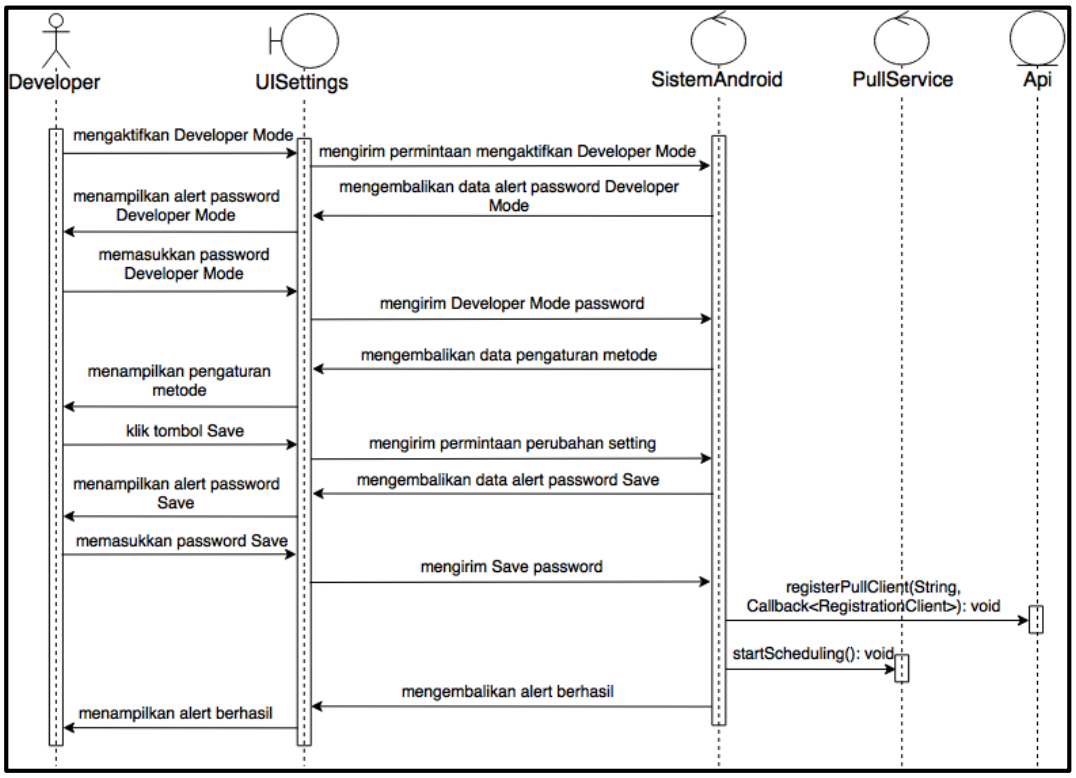


Gambar 3.10 Sequence Diagram Mengubah Metode Notifikasi Menjadi Push

Gambar 3.11 merupakan Sequence Diagram Mengubah Metode Notifikasi Menjadi Pull. Skenario untuk *event* tersebut:

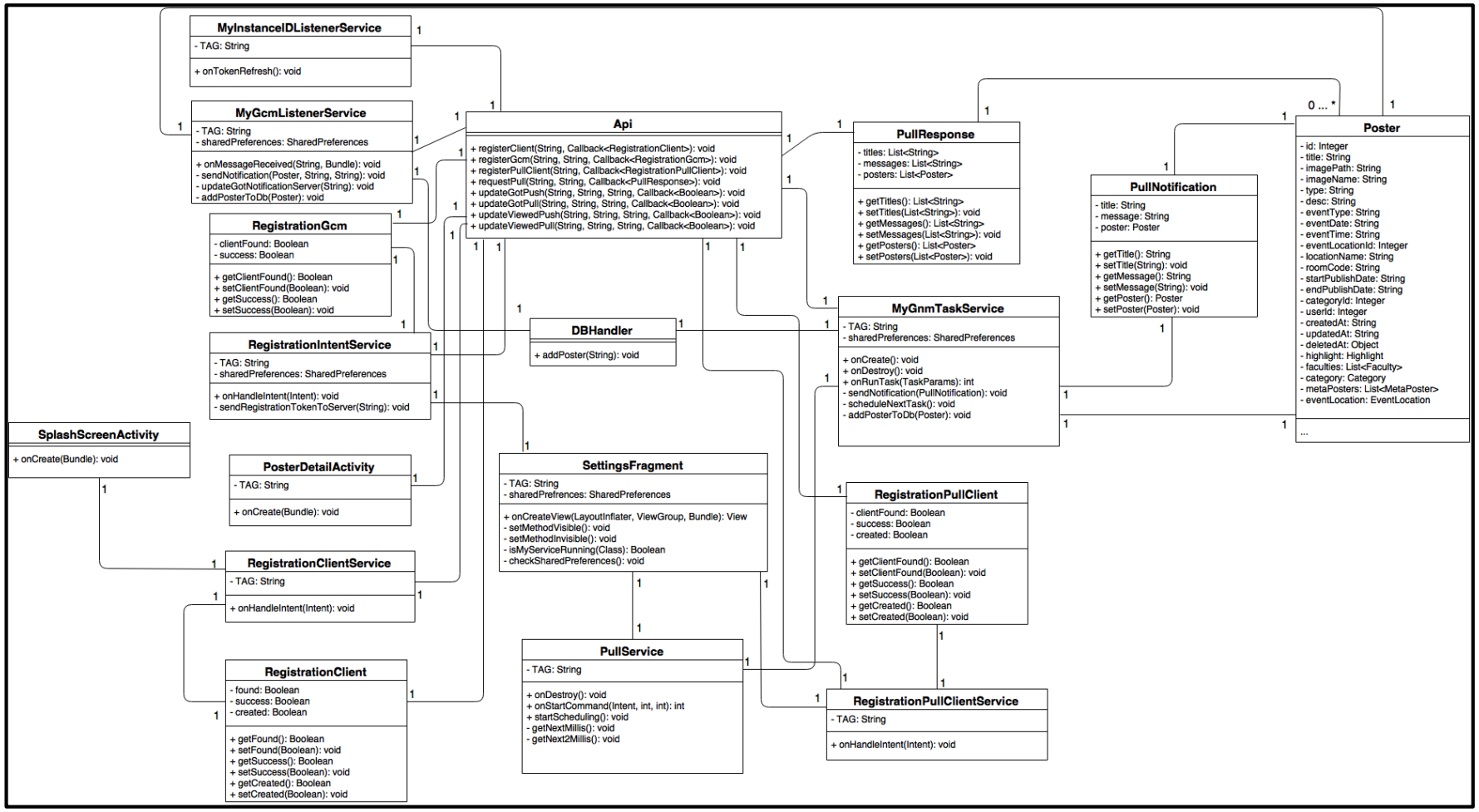
1) Developer yang berada pada UISettings, mengubah pilihan Developer Mode.
2) Selanjutnya Developer akan ditampilkan alert oleh UI untuk memasukkan *password* Developer Mode.
3) Kemudian UI akan menampilkan pilihan Method, dengan *default* Method adalah Pull.
4) Developer menekan tombol Save.
5) UI menampilkan *alert* untuk memasukkan *password* Save.
6) Developer memasukkan *password* Save.

7) Sistem Android akan mendaftarkan ID perangkat pengguna sebagai pengguna aplikasi *pull* dan menjadwalkan *request* selanjutnya dengan memanggil fungsi startScheduling pada PullService.



Gambar 3.11 Sequence Diagram Mengubah Metode Notifikasi Menjadi Pull

Tahap perancangan selanjutnya adalah pembuatan *Class Diagram* untuk mengetahui *class* yang digunakan beserta dengan hubungan antar *class* tersebut. Gambar 3.12 menjelaskan hubungan antar *class* yang digunakan pada Aplikasi UMN Bulletin.



Gambar 3.12 Class Diagram Aplikasi UMN Bulletin

Terdapat tujuh belas *class*, dengan dua *class* sebagai Activity, satu *class* sebagai Fragment, tujuh *class* sebagai Service, enam *class* sebagai Model, dan satu *class* yang digunakan untuk berkomunikasi dengan API UMN Bulletin.

Class yang berfungsi sebagai Activity adalah SplashScreenActivity dan PosterDetailActivity. *Class* yang berfungsi sebagai Fragment adalah SettingsFragment. *Class* yang berfungsi sebagai Service adalah RegistrationClientService, MyInstanceIDListenerService, MyGcmListenerService, RegistrationIntentService, RegistrationPullClient, PullService, dan MyGnmTaskService. *Class* yang berfungsi sebagai Model adalah RegistrationClient, RegistrationGcm, RegistrationPullClient, PullResponse, PullNotification, dan Poster. *Class* Api bertujuan untuk berkomunikasi dengan API dengan menggunakan *networking framework* Retrofit.

Pada *class* SplashScreenActivity akan dilakukan pendaftaran ID perangkat pengguna dengan menggunakan RegistrationClientService. Sedangkan pada *class* PosterDetailActivity, waktu notifikasi dibuka akan disampaikan ke *server* dengan berhubungan langsung dengan *class* Api. SettingsFragment merupakan sebuah *fragment* yang digunakan untuk memilih metode penerimaan notifikasi pada aplikasi.

RegistrationClientService merupakan sebuah *service* yang berhubungan langsung dengan *class* Api untuk mendaftarkan ID perangkat pengguna ke *server*. Sama halnya dengan RegistrationIntentService dan RegistrationPullClientService, kedua *service* ini juga berhubungan langsung dengan *class* Api untuk mendaftarkan sebagai pengguna *push* (RegistrationIntentService) atau *pull* (RegistrationPullClientService). MyInstanceIDListenerService merupakan sebuah

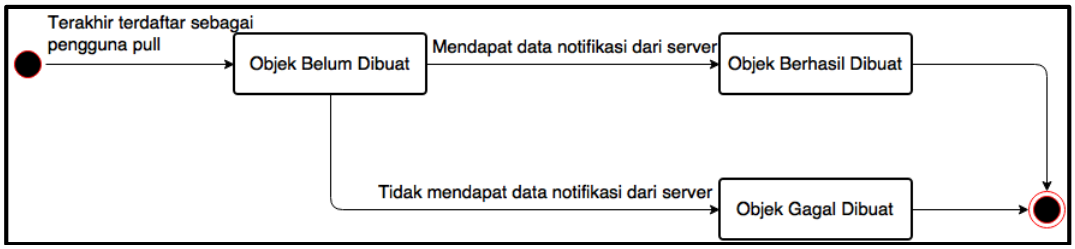
service GCM yang bertujuan untuk mengontrol apabila ada perubahan token registrasi, yaitu meng-*update* data pada *server* dengan cara berhubungan dengan *class* Api. MyGcmListenerService merupakan sebuah *service GCM* yang akan berjalan secara otomatis apabila terdapat notifikasi yang didapat dari *server* GCM. *Service* ini akan meng-*update* waktu notifikasi diterima. PullService merupakan *service* yang digunakan untuk menjadwalkan *request* selanjutnya. MyGnmTaskService merupakan sebuah *service* yang akan dijalankan ketika *request* penjadwalan akan dilakukan. *Service* ini berhubungan dengan *class* Api untuk mengambil data poster terbaru dan meng-*update* waktu notifikasi diterima.

Model RegistrationClient, RegistrationGcm, RegistrationPullClient, dan PullResponse berhubungan dengan *class* Api karena mengikuti *framework* Retrofit, yaitu respon dari API yang diterima akan otomatis berupa objek *model*, contoh pemanggilan fungsi registerClient pada *class* Api akan menghasilkan sebuah objek RegistrationClient. PullResponse merupakan *model* yang berisi daftar judul, pesan, dan poster yang diterima setelah melakukan permintaan pengambilan data poster baru. Untuk mempermudah membuat notifikasi, dibuat sebuah *model* bernama PullNotification yang hanya berisi satu judul, pesan, dan poster. Poster merupakan *model* yang telah ada pada aplikasi, yang berhubungan dengan PullResponse dan PullNotification dan digunakan oleh MyGcmListenerService.

Terdapat sebuah variabel bernama sharedPreferences yang memiliki tipe data SharedPreferences pada beberapa *class* yang digunakan. Variabel ini berfungsi untuk menyimpan status aplikasi mengenai pendaftaran pengguna terakhir. Misal, pada minggu pertama pengguna terdaftar sebagai pengguna *push* maka statusnya adalah sebagai pengguna *push*. Kemudian, pada minggu kedua pengguna terdaftar

sebagai pengguna *pull* maka statusnya adalah sebagai pengguna *pull*.

Class PullResponse dapat berisi nol atau banyak Poster. Hal ini dikarenakan apabila data notifikasi yang diterima dari *server* UMN Bulletin tidak ada (*null*), maka PullResponse berisi nol Poster. Ketiadaaan data notifikasi yang diterima oleh aplikasi dapat disebabkan karena memang tidak ada data poster baru yang belum pernah diterima atau koneksi internet pada perangkat yang tiba-tiba menghilang. Hal ini hanya dapat terjadi pada metode *pull* karena metode *pull* membutuhkan koneksi internet pada dua tahap, yaitu pada saat *request* dan mendapat *response*, sedangkan untuk mendapatkan data notifikasi pada metode *push* hanya membutuhkan satu tahap, dimana data notifikasi akan langsung dikirimkan dari *server* GCM segera setelah koneksi internet pada perangkat ada. Gambar 3.13 merupakan State Chart Diagram yang menjelaskan perubahan status dari objek Poster.



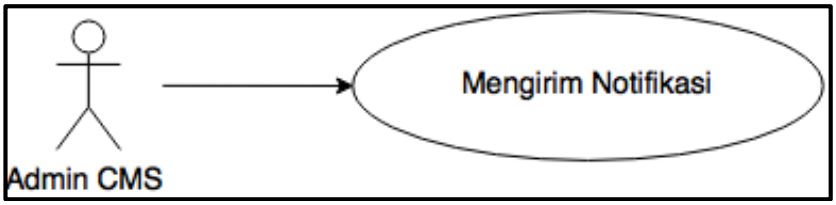
Gambar 3.13 State Chart Diagram Objek Poster Aplikasi

Seperti yang telah dijelaskan pada Gambar 3.13, status objek akan menjadi belum dibuat ketika status aplikasi mengenai pendaftaran pengguna terakhir adalah sebagai pengguna *pull*. Dengan status aplikasi sebagai pengguna *pull*, aplikasi akan melakukan penjadwalan permintaan data poster terbaru ke *server* UMN Bulletin. Kemudian status objek Poster akan berubah menjadi berhasil dibuat apabila mendapatkan data notfikasi dari *server* UMN Bulletin. Namun apabila tidak mendapatkan data notifikasi, status objek Poster akan berubah menjadi gagal

dibuat. Untuk mendapatkan data notifikasi dari *server* UMN Buletin, pengguna harus memiliki koneksi internet pada saat menerima *response* dan terdapat data poster baru yang belum pernah diterima oleh pengguna. Namun, ketika pengguna tidak memiliki koneksi internet pada saat menerima *response* atau tidak terdapat data poster baru yang belum pernah diterima pengguna maka aplikasi tidak akan mendapatkan data notifikasi dari *server* UMN Bulletin, atau dengan kata lain objek Poster akan gagal dibuat.

3.2.2 Perancangan CMS UMN Bulletin

Sama seperti perancangan Aplikasi UMN Bulletin, tahap pertama yang dilakukan untuk merancang CMS UMN Bulletin adalah dengan membuat *Use Case Diagram*. Gambar 3.14 merupakan *use case diagram* untuk CMS UMN Bulletin.



Gambar 3.14 Use Case Diagram CMS UMN Bulletin

Use Case Mengirim Notifikasi dijelaskan pada Tabel 3.8.

Tabel 3.8 Use Case Description Mengirim Notifikasi

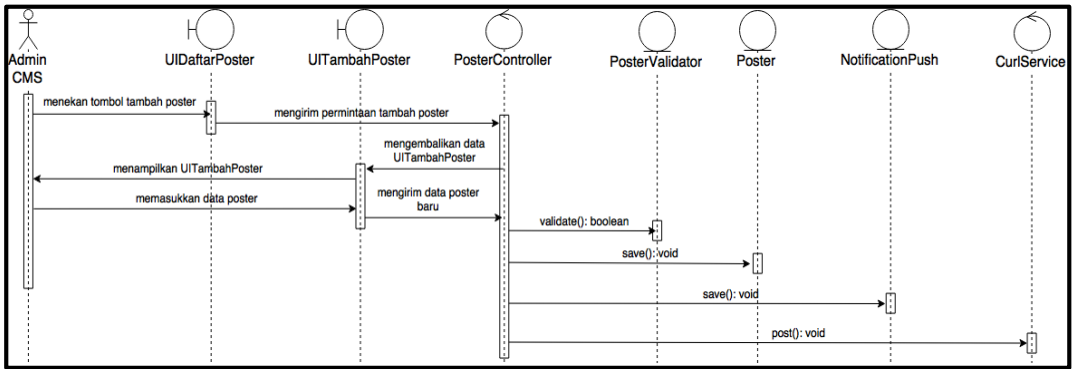
Use Case Name	Mengirim Notifikasi
Actor	Admin CMS
Description	Merupakan sebuah *event* dimana aktor mengirimkan notifikasi pada pengguna aplikasi UMN Bulletin dengan menambahkan data poster baru ke CMS UMN Bulletin
Trigger	Aktor ingin menambahkan data poster baru ke CMS UMN Bulletin
Normal Flow of Event	1. Aktor menambahkan data poster baru ke CMS UMN Bulletin. 2. CMS UMN Bulletin akan menambahkan data poster baru tersebut ke database.

Tabel 3.8 Use Case Description Mengirim Notifikasi (Lanjutan)

Normal Flow of Event	3. Selanjutnya, CMS UMN Bulletin akan mengirimkan data poster baru beserta dengan token registrasi para pengguna aplikasi *push* ke server GCM.
Pre Condition	-
Post Condition	Aktor berhasil menambahkan data poster baru ke database server dan mengirimkan notifikasi ke para pengguna aplikasi *push*

Tahap selanjutnya dalam mengembangkan CMS UMN Bulletin adalah dengan membuat *Sequence Diagram*. Gambar 3.15 merupakan Sequence Diagram Mengirim Notifikasi. Skenario untuk *event* tersebut:

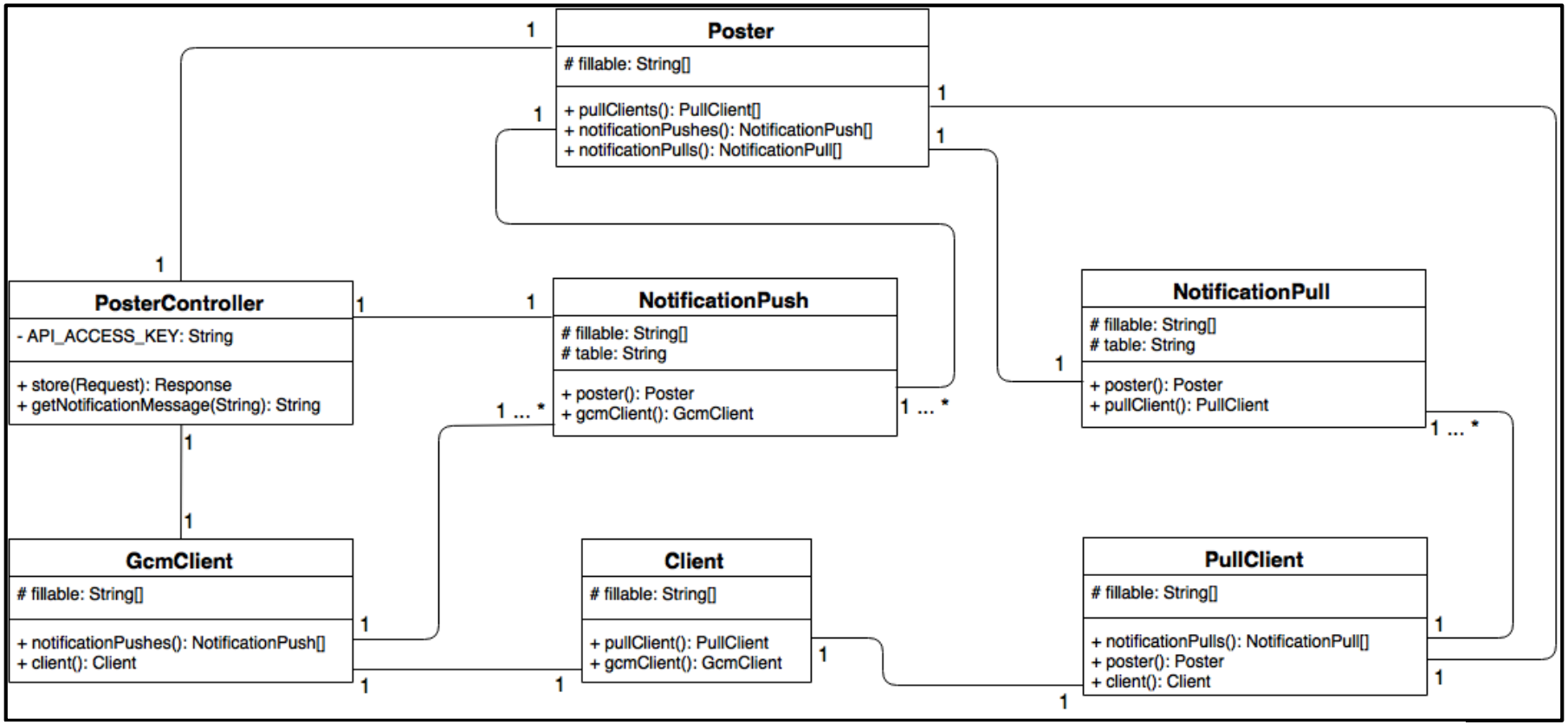
1) Aktor berada pada laman daftar poster, menekan tombol tambah poster.
2) UI akan meneruskan permintaan aktor ke PosterController (*controller*).
3) *Controller* mengirimkan data berupa laman tambah poster ke UI, kemudian UI menampilkannya ke aktor.
4) Aktor memasukkan data-data poster baru ke UI untuk diteruskan ke *controller*.
5) *Controller* akan memvalidasi data dengan memanggil fungsi validate() pada PosterValidator
6) Controller kemudian menyimpan poster baru melalui model Poster dan data-data notifikasi yang akan dikirimkan ke NotificationPush dengan fungsi save().
7) *Controller* akan mengirimkan data-data notifikasi ke *server* GCM dengan menggunakan *custom* PHP cURL *library* CurlService dengan fungsi post().



Gambar 3.15 Sequence Diagram Mengirim Notifikasi

Selanjutnya, untuk menjelaskan *class* yang digunakan beserta hubungannya, dibuat suatu *Class Diagram*. *Class* yang digunakan dalam CMS UMN Bulletin terdiri dari tujuh *class*. Dari ketujuh *class* tersebut, enam diantaranya merupakan *model* dan satu *class controller*. Keenam *model* tersebut adalah Poster, Client, GcmClient, PullClient, NotificationPush, dan NotificationPull. Sedangkan *controller* yang digunakan bernama PosterController.

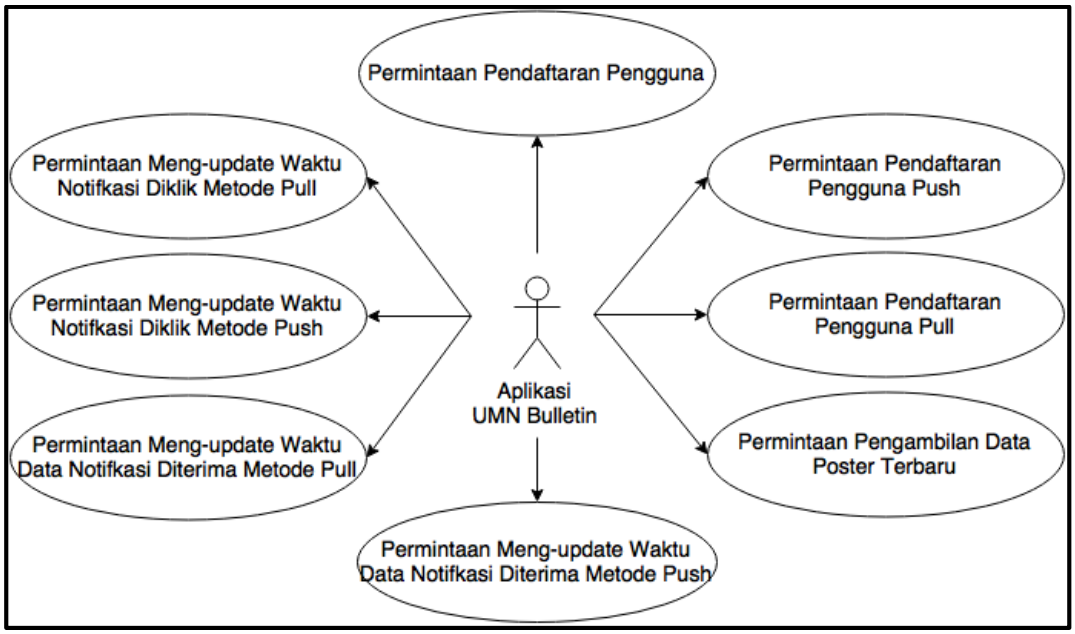
Aktivitas menambahkan poster baru akan dikontrol oleh PosterController dengan fungsi store(Request). Pada fungsi inilah *controller* akan berhubungan dengan *model* Poster, NotificationPush, dan GcmClient yang bertujuan untuk menambahkan data poster baru dan data notifikasi yang dikirimkan. Pada fungsi ini pula data notifikasi dikirimkan ke *server* GCM untuk diteruskan ke para pengguna aplikasi *push*. Gambar 3.16 menjelaskan *class* yang digunakan dalam CMS UMN Bulletin beserta dengan hubungan antar *class* tersebut.



Gambar 3.16 Class Diagram Content Management System UMN Bulletin

3.2.3 Perancangan API UMN Bulletin

Tahap awal dalam perancangan API UMN Bulletin adalah dengan membuat *Use Case Diagram*. Aktor yang terdapat pada API UMN Bulletin adalah sistem di luar API UMN Bulletin sendiri, yaitu Aplikasi UMN Bulletin. Gambar 3.17 merupakan *use case diagram* untuk API UMN Bulletin.



Gambar 3.17 Use Case Diagram API UMN Bulletin

Use Case Diagram Permintaan Pendaftaran Pengguna dijelaskan pada Tabel 3.9.

Tabel 3.9 Use Case Description Permintaan Pendaftaran Pengguna

Use Case Name	Permintaan Pendaftaran Pengguna
Actor	Aplikasi UMN Bulletin
Description	Merupakan sebuah *event* dimana aktor melakukan permintaan untuk mendaftar sebagai pengguna aplikasi
Trigger	Aktor ingin mendaftarkan pengguna sebagai pengguna aplikasi
Normal Flow of Event	1. Aktor mengirimkan permintaan untuk mendaftar sebagai pengguna aplikasi ke API UMN Bulletin.

Tabel 3.9 Use Case Description Permintaan Pendaftaran Pengguna (Lanjutan)

Normal Flow of Event	2. Apabila pengguna belum ada, maka API UMN Bulletin akan menambahkan ID perangkat pengguna baru ke tabel clients, kemudian mengembalikan *response* keberhasilan proses penyimpanan. 3. Apabila pengguna sudah ada, maka API UMN Bulletin akan langsung mengembalikan *response* bahwa pengguna aplikasi sudah terdaftar.
Pre Condition	-
Post Condition	Aktor berhasil menambahkan data pengguna aplikasi baru

Use Case Diagram Permintaan Pendaftaran Pengguna Push dijelaskan pada Tabel 3.10.

Tabel 3.10 Use Case Description Permintaan Pendaftaran Pengguna Push

Use Case Name	Permintaan Pendaftaran Pengguna Push
Actor	Aplikasi UMN Bulletin
Description	Merupakan sebuah *event* dimana aktor melakukan permintaan untuk mendaftar sebagai pengguna aplikasi *push*
Trigger	Aktor ingin mendaftarkan pengguna sebagai pengguna aplikasi *push*
Normal Flow of Event	1. Aktor mengirimkan permintaan untuk mendaftar sebagai pengguna aplikasi *push* ke API UMN Bulletin. 2. Apabila pengguna *push* belum ada, maka API UMN Bulletin akan menambahkan ID perangkat pengguna beserta dengan token registrasinya ke tabel gcm_clients, kemudian mengembalikan *response* keberhasilan proses penyimpanan. 3. Apabila pengguna *push* sudah ada, maka API UMN Bulletin akan meng-*update* token registrasi yang terdapat pada tabel gcm_clients kemudian mengembalikan *response* keberhasilan proses penyimpanan.

Tabel 3.10 Use Case Description Permintaan
Pendaftaran Pengguna Push (Lanjutan)

Pre Condition	-
Post Condition	Aktor berhasil menambahkan data pengguna aplikasi *push* baru atau meng-*update* token registrasi pengguna pada server

Use Case Diagram Permintaan Pendaftaran Pengguna Pull dijelaskan pada Tabel 3.11.

Tabel 3.11 Use Case Description Permintaan Pendaftaran Pengguna Pull

Use Case Name	Permintaan Pendaftaran Pengguna Pull
Actor	Aplikasi UMN Bulletin
Description	Merupakan sebuah *event* dimana aktor melakukan permintaan untuk mendaftar sebagai pengguna aplikasi *pull*
Trigger	Aktor ingin mendaftarkan pengguna sebagai pengguna aplikasi *pull*
Normal Flow of Event	1. Aktor mengirimkan permintaan untuk mendaftar sebagai pengguna aplikasi *pull* ke API UMN Bulletin. 2. Apabila pengguna *pull* belum ada, maka API UMN Bulletin akan menambahkan ID perangkat pengguna ke tabel pull_clients, kemudian mengembalikan *response* keberhasilan proses penyimpanan. 3. Apabila pengguna *pull* sudah ada, maka API UMN Bulletin akan langsung mengembalikan *response* bahwa pengguna aplikasi *pull* sudah terdaftar.
Pre Condition	-
Post Condition	Aktor berhasil menambahkan data pengguna aplikasi *pull* baru

Use Case Diagram Permintaan Pengambilan Data Poster Terbaru dijelaskan pada Tabel 3.12.

Tabel 3.12 Use Case Description Permintaan
Pengambilan Data Poster Terbaru

Use Case Name	Permintaan Pengambilan Data Poster Terbaru
Actor	Aplikasi UMN Bulletin

Tabel 3.12 Use Case Description Permintaan Pengambilan Data Poster Terbaru (Lanjutan)

Description	Merupakan sebuah *event* dimana aktor melakukan permintaan untuk mengambil data poster terbaru pada server
Trigger	Aktor ingin mengambil data poster terbaru
Normal Flow of Event	1. Aktor mengirimkan permintaan untuk mengambil data poster terbaru ke API UMN Bulletin. 2. Apabila terdapat data poster baru yang belum pernah diterima oleh aktor, maka API UMN Bulletin akan mencatat waktu melakukan permintaan pada tabel notification_pulled dan memberikan *response* berupa data poster-poster yang belum pernah diterima aktor. 3. Apabila data poster baru sudah pernah diterima oleh aktor, maka API UMN Bulletin akan memberikan *response* berupa *null*.
Pre Condition	-
Post Condition	Aktor berhasil mendapatkan data notifikasi yang berupa data poster baru

Use Case Diagram Permintaan Meng-*update* Waktu Data Notifikasi Diterima Metode Push dijelaskan pada Tabel 3.13.

Tabel 3.13 Use Case Description Permintaan Meng-update Waktu Data Notifikasi Diterima Metode Push

Use Case Name	Permintaan Meng-update Waktu Data Notifikasi Diterima Metode Push
Actor	Aplikasi UMN Bulletin
Description	Merupakan sebuah *event* dimana aktor melakukan permintaan untuk meng-*update* waktu data notifikasi diterima untuk metode *push*
Trigger	Aktor ingin meng-*update* waktu data notifikasi diterima untuk metode *push*
Normal Flow of Event	1. Aktor mengirimkan permintaan meng-*update* waktu data notifikasi diterima untuk metode *push*. 2. API UMN Bulletin meng-*update* waktu data notifikasi diterima berdasarkan ID perangkat pengguna dan ID poster pada tabel notification_pushed.

Tabel 3.13 Use Case Description Permintaan Meng-update Waktu Data Notifikasi Diterima Metode Push (Lanjutan)

Normal Flow of Event	3. API UMN Bulletin memberikan *response* berupa pesan keberhasilan penyimpanan.
Pre Condition	-
Post Condition	Aktor berhasil meng-*update* waktu data notifikasi diterima untuk metode *push*

Use Case Diagram Permintaan Meng-*update* Waktu Data Notifikasi Diterima Metode Pull dijelaskan pada Tabel 3.14.

Tabel 3.14 Use Case Description Permintaan Meng-update Waktu Data Notifikasi Diterima Metode Pull

Use Case Name	Permintaan Meng-update Waktu Data Notifikasi Diterima Metode Pull
Actor	Aplikasi UMN Bulletin
Description	Merupakan sebuah *event* dimana aktor melakukan permintaan untuk meng-*update* waktu data notifikasi diterima untuk metode *pull*
Trigger	Aktor ingin meng-*update* waktu data notifikasi diterima untuk metode *pull*
Normal Flow of Event	1. Aktor mengirimkan permintaan meng-*update* waktu data notifikasi diterima untuk metode *pull*. 2. API UMN Bulletin meng-*update* waktu data notifikasi diterima berdasarkan ID perangkat pengguna dan ID poster pada tabel notification_pulled kemudian memberikan *response* berupa pesan keberhasilan penyimpanan.
Pre Condition	-
Post Condition	Aktor berhasil meng-*update* waktu data notifikasi diterima untuk metode *pull*

Use Case Diagram Permintaan Meng-*update* Waktu Notifikasi Diklik Metode Push dijelaskan pada Tabel 3.15.

Tabel 3.15 Use Case Description Permintaan Meng-update Waktu Notifikasi Diklik Metode Push

Use Case Name	Permintaan Meng-update Waktu Notifikasi Diklik Metode Push
Actor	Aplikasi UMN Bulletin
Description	Merupakan sebuah *event* dimana aktor melakukan permintaan untuk meng-*update* waktu notifikasi diklik untuk metode *push*
Trigger	Aktor ingin meng-*update* waktu notifikasi diklik untuk metode *push*
Normal Flow of Event	1. Aktor mengirimkan permintaan untuk meng-*update* waktu notifikasi diklik untuk metode *push*. 2. API UMN Bulletin meng-*update* waktu notifikasi diklik berdasarkan ID perangkat pengguna dan ID poster pada tabel notification_pushed. 3. API UMN Bulletin memberikan *response* berupa pesan keberhasilan penyimpanan.
Pre Condition	-
Post Condition	Aktor berhasil meng-*update* waktu notifikasi diklik untuk metode *push*

Use Case Diagram Permintaan Meng-*update* Waktu Notifikasi Diklik Metode Pull dijelaskan pada Tabel 3.16.

Tabel 3.16 Use Case Description Permintaan Meng-update Waktu Notifikasi Diklik Metode Pull

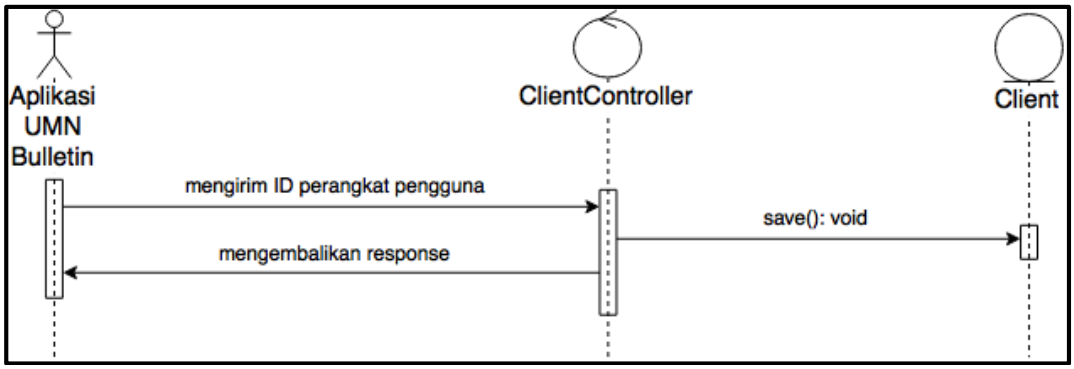
Use Case Name	Permintaan Meng-update Waktu Notifikasi Diklik Metode Pull
Actor	Aplikasi UMN Bulletin
Description	Merupakan sebuah *event* dimana aktor melakukan permintaan untuk meng-*update* waktu notifikasi diklik untuk metode *pull*
Trigger	Aktor ingin meng-*update* waktu notifikasi diklik untuk metode *pull*
Normal Flow of Event	1. Aktor mengirimkan permintaan untuk meng-*update* waktu notifikasi diklik untuk metode *pull*. 2. API UMN Bulletin meng-*update* waktu notifikasi diklik berdasarkan ID perangkat dan ID poster pada tabel notification_pulled.

Tabel 3.16 Use Case Description Permintaan Meng-update Waktu Notifikasi Diklik Metode Pull (Lanjutan)

Normal Flow of Event	3. API UMN Bulletin memberikan *response* berupa pesan keberhasilan penyimpanan.
Pre Condition	-
Post Condition	Aktor berhasil meng-*update* waktu notifikasi diklik untuk metode *pull*

Tahap selanjutnya adalah pembuatan *Sequence Diagram*. Gambar 3.18 merupakan Sequence Diagram Permintaan Pendaftaran Pengguna. Skenario untuk *event* tersebut:

1) Aktor mengirimkan permintaan pendaftaran pengguna beserta dengan ID perangkat pengguna ke API UMN Bulletin, yang kemudian diterima oleh *controller* ClientController.
2) *Controller* menyimpan ID perangkat pengguna melalui *model* Client.
3) *Controller* mengembalikan *response* berupa keberhasilan proses penyimpanan.



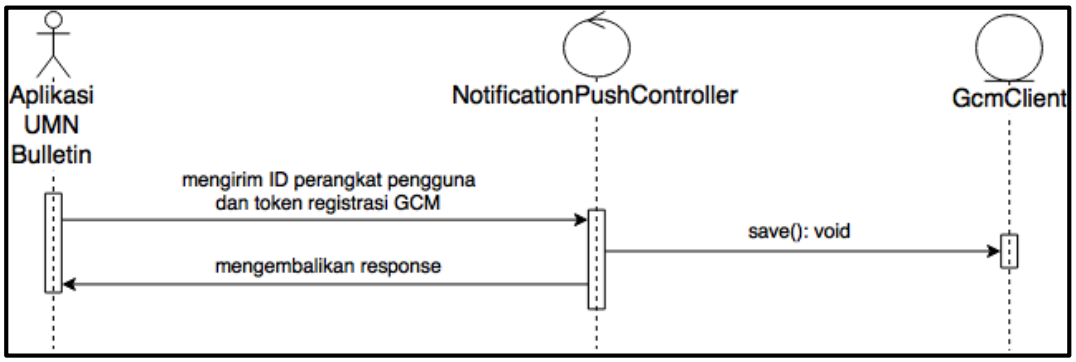
Gambar 3.18 Sequence Diagram Permintaan Pendaftaran Pengguna

Gambar 3.19 merupakan Sequence Diagram Permintaan Pendaftaran Pengguna Push. Skenario untuk *event* tersebut:

1) Aktor mengirimkan permintaan pendaftaran pengguna *push* beserta dengan ID perangkat pengguna dan token registrasi GCM-nya ke API UMN Bulletin, yang kemudian diterima oleh *controller* NotificationPushController.

2) *Controller* menyimpan data pengguna *push* melalui *model* GcmClient.

3) *Controller* mengembalikan *response* berupa keberhasilan proses penyimpanan.



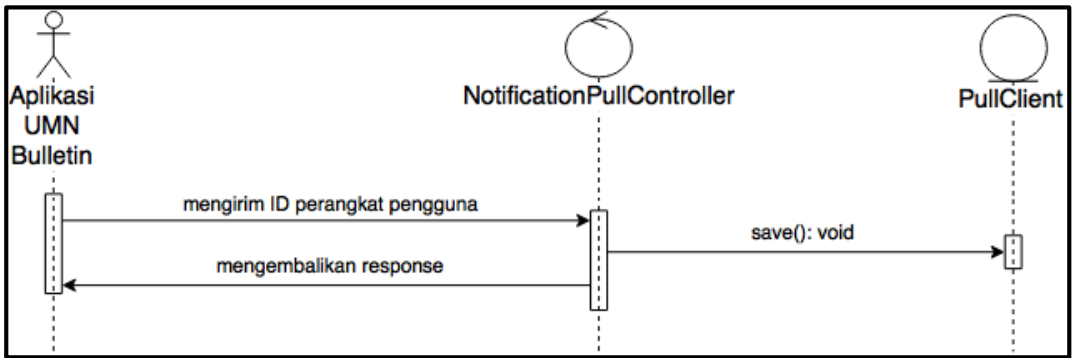
Gambar 3.19 Sequence Diagram Permintaan Pendaftaran Pengguna Push

Gambar 3.20 merupakan Sequence Diagram Permintaan Pendaftaran Pengguna Pull. Skenario untuk *event* tersebut:

1) Aktor mengirimkan permintaan pendaftaran pengguna *pull* beserta dengan ID perangkat pengguna ke API UMN Bulletin, yang kemudian diterima oleh *controller* NotificationPullController.

2) *Controller* menyimpan data pengguna *pull* melalui *model* PullClient.

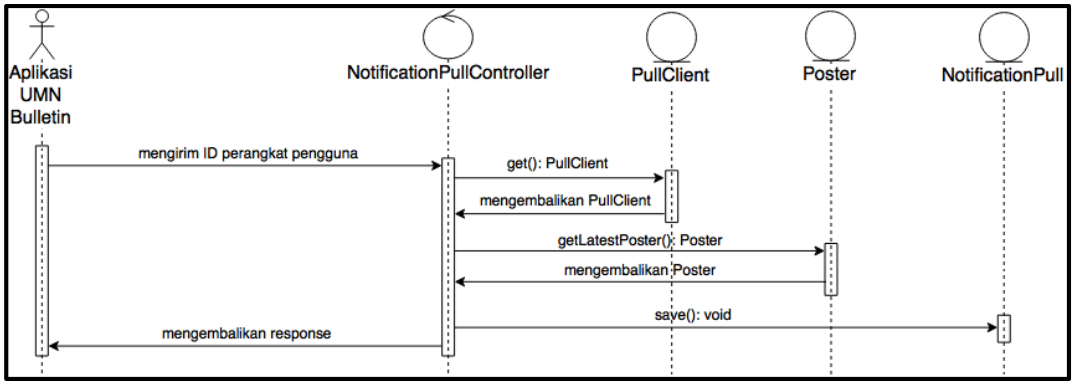
3) *Controller* mengembalikan *response* berupa keberhasilan proses penyimpanan.



Gambar 3.20 Sequence Diagram Permintaan Pendaftaran Pengguna Pull

Gambar 3.21 merupakan Sequence Diagram Permintaan Pengambilan Data Poster Terbaru. Skenario untuk *event* tersebut:

1) Aktor mengirimkan permintaan pengambilan data poster terbaru beserta dengan ID perangkat pengguna ke API UMN Bulletin, yang kemudian diterima oleh *controller* NotificationPullController.
2) *Controller* mengambil data pengguna *pull* melalui *model* PullClient.
3) *Controller* mengambil data poster terakhir melalui *model* Poster.
4) Proses selanjutnya yang dilakukan di dalam *controller* adalah mengecek ID poster terakhir yang diterima pengguna dengan ID poster terakhir. Apabila pengguna belum mendapatkan data poster terakhir tersebut maka *controller* akan menyimpan data notifikasi yang akan dikirimkan menjadi *response* ke tabel NotificationPull.

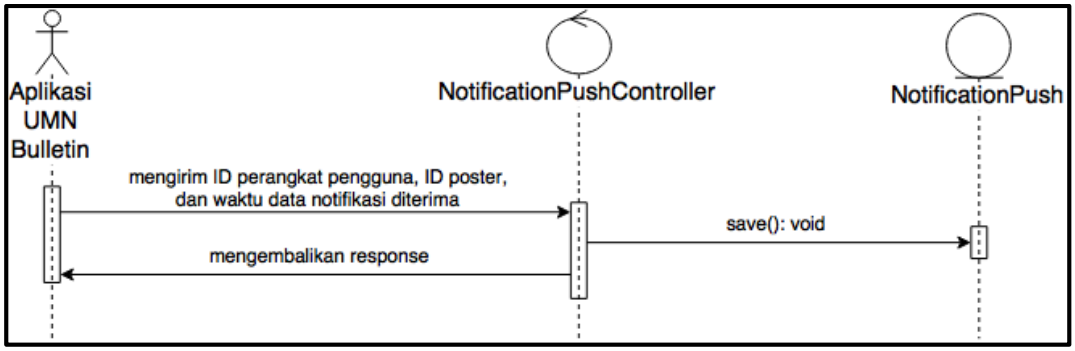


Gambar 3.21 Sequence Diagram Permintaan Pengambilan Data Poster Terbaru

Gambar 3.22 merupakan Sequence Diagram Permintaan Meng-*update* Waktu Data Notifikasi Diterima Metode Push. Skenario untuk *event* tersebut:

1) Aktor mengirimkan permintaan meng-*update* data waktu notifikasi diterima untuk metode *push* beserta dengan ID perangkat pengguna, ID poster, dan waktu data notifikasi diterima ke API UMN Bulletin, yang kemudian diterima oleh *controller* NotificationPushController.
2) *Controller* meng-*update* waktu data notifikasi diterima berdasarkan ID perangkat pengguna dan ID poster melalui *model* NotificationPush.

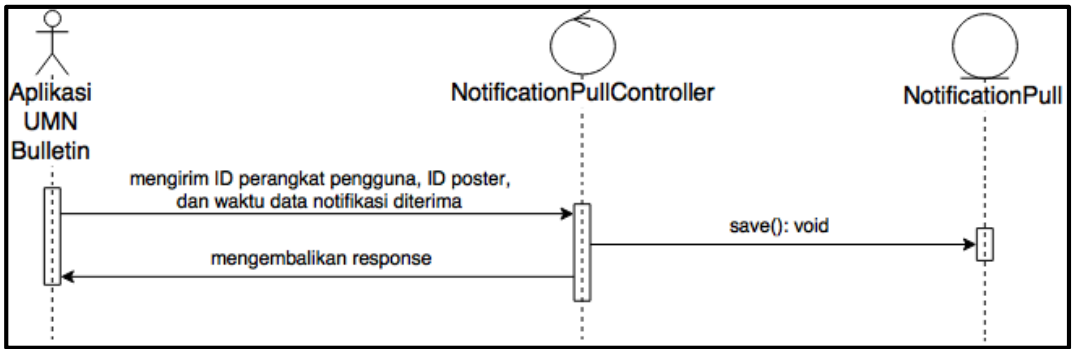
3) *Controller* mengembalikan *response* berupa keberhasilan proses penyimpanan.



Gambar 3.22 Sequence Diagram Permintaan Meng-update Waktu Data Notifikasi Diterima Metode Push

Gambar 3.23 merupakan Sequence Diagram Permintaan Meng-*update* Waktu Data Notifikasi Diterima Metode Pull. Skenario untuk *event* tersebut:

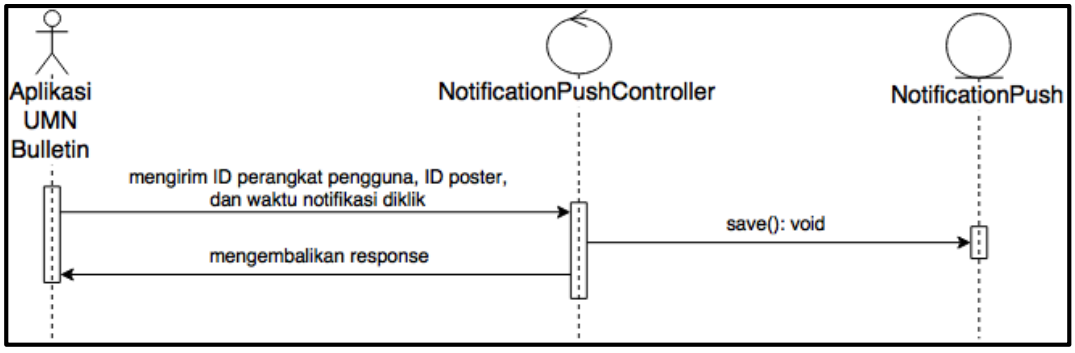
1) Aktor mengirimkan permintaan meng-*update* data waktu notifikasi diterima untuk metode *pull* beserta dengan ID perangkat pengguna, ID poster, dan waktu data notifikasi diterima ke API UMN Bulletin, yang kemudian diterima oleh *controller* NotificationPullController.
2) *Controller* meng-*update* waktu data notifikasi diterima berdasarkan ID perangkat pengguna dan ID poster melalui *model* NotificationPull.
3) *Controller* mengembalikan *response* berupa keberhasilan proses penyimpanan.



Gambar 3.23 Sequence Diagram Permintaan Meng-update Waktu Data Notifikasi Diterima Metode Pull

Gambar 3.24 merupakan Sequence Diagram Permintaan Meng-*update* Waktu Notifikasi Diklik Metode Push. Skenario untuk *event* tersebut:

1) Aktor mengirimkan permintaan meng-*update* waktu notifikasi diklik untuk metode *push* beserta dengan ID perangkat pengguna, ID poster, dan waktu notifikasi diklik ke API UMN Bulletin, yang kemudian diterima oleh *controller* NotificationPushController.
2) *Controller* meng-*update* waktu notifikasi diklik berdasarkan ID perangkat pengguna dan ID poster melalui *model* NotificationPush.
3) *Controller* mengembalikan *response* berupa keberhasilan proses penyimpanan.

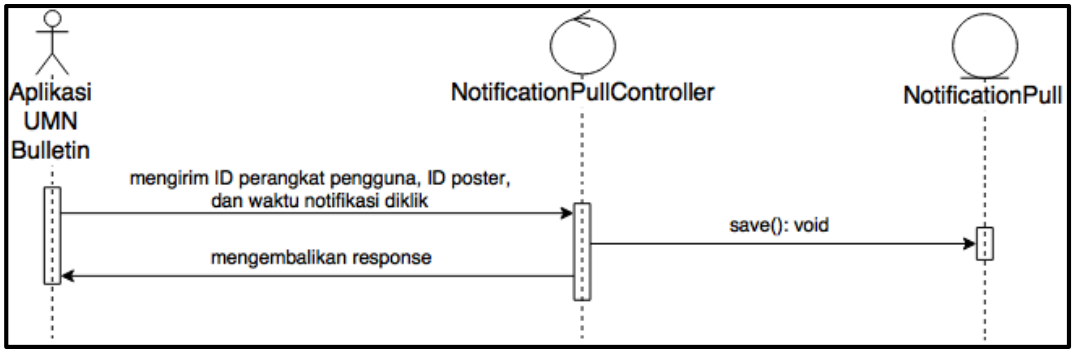


Gambar 3.24 Sequence Diagram Permintaan Meng-update Waktu Notifikasi Diklik Metode Push

Gambar 3.25 merupakan Sequence Diagram Permintaan Meng-*update* Waktu Notifikasi Diklik Metode Pull. Skenario untuk *event* tersebut:

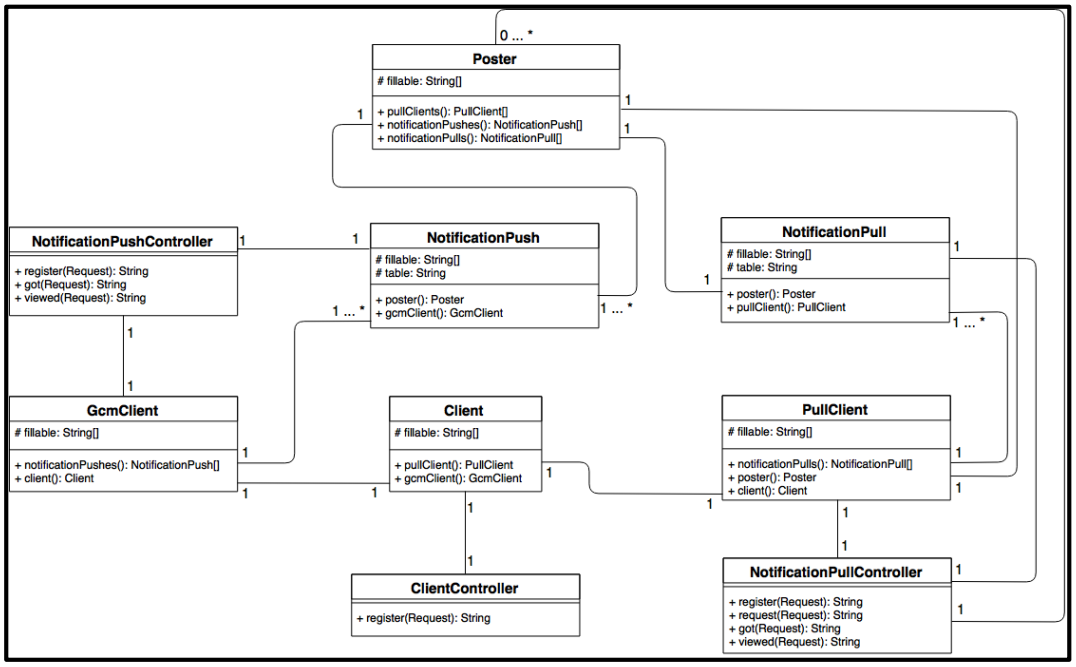
1) Aktor mengirimkan permintaan meng-*update* waktu notifikasi diklik untuk metode *pull* beserta dengan ID perangkat pengguna, ID poster, dan waktu data notifikasi diterima ke API UMN Bulletin, yang kemudian diterima oleh *controller* NotificationPullController.
2) *Controller* meng-*update* waktu notifikasi diklik berdasarkan ID perangkat pengguna dan ID poster melalui *model* NotificationPull.

3) *Controller* mengembalikan *response* berupa keberhasilan proses penyimpanan.



Gambar 3.25 Sequence Diagram Permintaan Meng-update Waktu Notifikasi Diklik Metode Pull

Setelah membuat *Sequence Diagram*, dibuat sebuah *Class Diagram* untuk mengetahui *class* yang digunakan oleh API UMN Bulletin beserta dengan hubungan antar *class* tersebut. Terdapat sembilan *class* yang digunakan oleh API UMN Bulletin, yang terdiri dari tiga *controller* dan enam *model*. Tiga *class* yang berperan sebagai *controller* adalah ClientController, NotificationPushController, dan NotificationPullController, sedangkan enam *class* lainnya yang berperan sebagai *model* adalah Poster, Client, GcmClient, PullClient, NotificationPush, dan NotificationPull. Gambar 3.24 menjelaskan hubungan antar *class* tersebut.

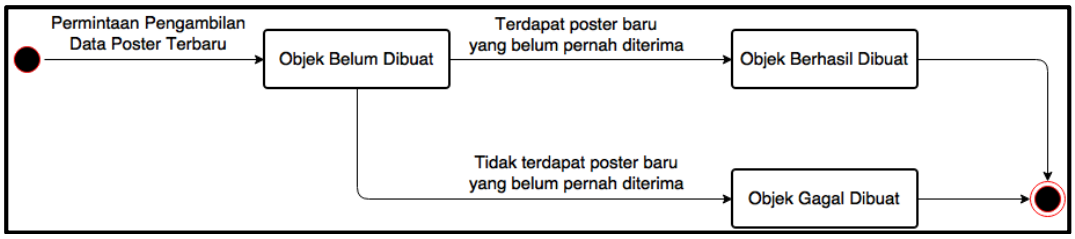


Gambar 3.26 Class Diagram API UMN Bulletin

Seperti yang telah dijelaskan pada Gambar 3.26, terdapat sebuah fungsi dengan nama yang sama pada ClientController, NotificationPushController, dan NotificationPullController, yaitu register(Request). Fungsi ini berfungsi untuk mendaftarkan pengguna sebagai pengguna aplikasi (ClientController), pengguna aplikasi *push* (NotificationPushController), atau sebagai pengguna aplikasi *pull* (NotificationPullController). Selain itu, terdapat dua nama fungsi lain yang sama pada NotificationPushController dan NotificationPullController, yaitu got(Request) dan viewed(Request). Fungsi got(Request) berfungsi untuk meng-*update* waktu data notifikasi diterima untuk metode *push* atau *pull*. Fungsi viewed(Request) berfungsi untuk meng-*update* waktu notifikasi diklik oleh pengguna untuk metode *push* atau *pull* yang ditentukan dengan *controller* mana yang memanggil fungsi tersebut.

Fungsi request(Request) yang terdapat pada NotificationPullController berfungsi untuk menerima permintaan pengambilan data poster terbaru dari aplikasi

UMN Bulletin. Dalam fungsi inilah objek Poster tidak selalu dibuat, yang ditandai dengan hubungan 0 … *. Maka dari itu, dibuat sebuah *State Chart Diagram* yang menjelaskan status dari objek Poster. Gambar 3.27 merupakan State Chart Diagram yang menjelaskan perubahan status dari objek Poster.

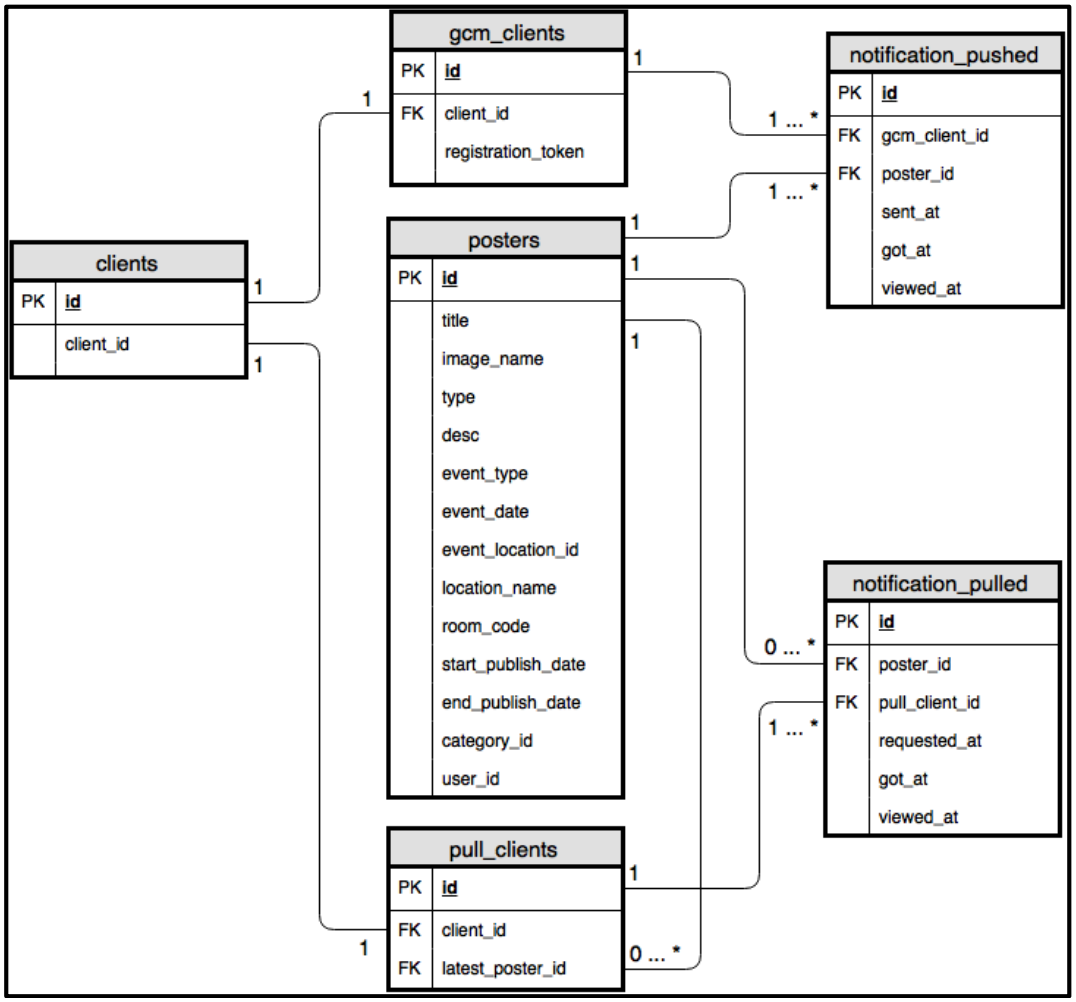


Gambar 3.27 State Chart Diagram Objek Poster API

Seperti yang telah dijelaskan sebelumnya, objek Poster tidak selalu berhasil dibuat. Hal ini dapat terjadi apabila pengguna aplikasi *pull* sudah menerima data poster terbaru, atau dengan kata lain tidak terdapat poster baru yang belum pernah diterima. Namun, apabila pengguna aplikasi *pull* belum pernah mendapat data poster terbaru, objek Poster akan berhasil dibuat.

3.2.4 Entity Relationship Diagram

Terdapat enam tabel yang digunakan dalam sistem, yaitu clients, gcm_clients, pull_clients, notification_pushed, notification_pulled, dan posters. Entitas posters merupakan entitas yang sudah ada pada sistem UMN Bulletin sebelumnya, sehingga hubungan entitas posters dengan entitas di luar sistem ini tidak digambarkan. Gambar 3.28 merupakan Entity Relationship Diagram (ERD) dari sistem UMN Bulletin.



Gambar 3.28 Entity Relationship Diagram

Tabel posters merupakan tabel untuk menyimpan data-data poster mading. *Primary key* untuk setiap tabel yang digunakan pada sistem ini adalah id. Selain id, Tabel ini terdiri dari kolom title untuk menyimpan judul poster, image_name untuk menyimpan nama gambar poster, type untuk menyimpan tipe poster, desc untuk menyimpan deskripsi poster, event_type untuk menyimpan apakah poster merupakan sebuah *event*, event_date untuk menyimpan tanggal *event*, event_location_id untuk menyimpan id lokasi *event*, location_name untuk menyimpan nama lokasi *event*, room_code untuk menyimpan ruangan *event*, start_publish_date untuk menyimpan tanggal mulai publikasi poster, end_publish_date untuk menyimpan tanggal selesai publikasi poster, category_id

untuk menyimpan id kategori poster, dan user_id untuk menyimpan id pengguna yang menambahkan poster. Struktur tabel posters dijelaskan pada Tabel 3.17.

Tabel 3.17 Struktur Tabel Posters

Nama Kolom	**Tipe Data**	**Panjang**
id (PK)	int	10
title	varchar	255
image_name	varchar	255
type	varchar	255
desc	longtext	-
event_type	varchar	255
event_date	date	-
event_location_id	time	-
location_name	varchar	255
room_code	varchar	255
start_publish_date	date	-
end_publish_date	date	-
category_id	int	10
user_id	int	10

Tabel clients merupakan tabel untuk menyimpan data pengguna aplikasi secara umum. ID perangkat pengguna aplikasi disimpan pada kolom client_id. Struktur tabel clients dijelaskan pada Tabel 3.18.

Tabel 3.18 Struktur Tabel Clients

Nama Kolom	**Tipe Data**	**Panjang**
id (PK)	bigint	20
client_id	varchar	255

Tabel gcm_clients merupakan tabel untuk menyimpan data pengguna aplikasi *push*. Pada tabel ini terdapat kolom registration_token yang berguna untuk menyimpan token registrasi GCM pengguna aplikasi *push*. Tabel ini memiliki *foreign key* ke tabel clients melalui client_id. Struktur tabel gcm_clients dijelaskan pada Tabel 3.19.

Tabel 3.19 Struktur Tabel Gcm_Clients

Nama Kolom	**Tipe Data**	**Panjang**
id (PK)	bigint	20
client_id (FK)	varchar	255
registration_token	varchar	255

Tabel pull_clients merupakan tabel untuk menyimpan data pengguna aplikasi *pull*. Latest_poster_id digunakan untuk menyimpan id poster terakhir yang telah didapat pengguna. Tabel ini memiliki *foreign key* ke tabel clients melalui client_id dan ke tabel posters melalui latest_poster_id. Struktur tabel pull_clients dijelaskan pada Tabel 3.20.

Tabel 3.20 Struktur Tabel Pull_Clients

Nama Kolom	**Tipe Data**	**Panjang**
id (PK)	bigint	20
client_id (FK)	varchar	255
latest_poster_id (FK)	int	11

Tabel notification_pushed merupakan tabel untuk menyimpan data-data notifikasi yang dikirimkan ke para pengguna *push*. Tabel ini terdiri dari kolom id sebagai *primary key*, gcm_client_id yang merupakan id pengguna aplikasi *push*, poster_id yang merupakan id poster yang dikirimkan, sent_at yang merupakan waktu data notifikasi dikirimkan, got_at yang merupakan waktu data notifikasi diterima oleh aplikasi UMN Bulletin, dan viewed_at yang merupakan waktu notifikasi diklik oleh pengguna aplikasi *push*. Struktur tabel notification_pushed dijelaskan pada Tabel 3.21.

Tabel 3.21 Struktur Tabel Notification_Pushed

Nama Kolom	**Tipe Data**	**Panjang**
id (PK)	bigint	20
gcm_client_id (FK)	bigint	20
poster_id (FK)	int	11
sent_at	datetime	-
got_at	datetime	-
viewed_at	datetime	-

Tabel notification_pulled merupakan tabel untuk menyimpan data-data notifikasi yang dikirimkan ke para pengguna *pull*. Tabel ini terdiri dari kolom id sebagai *primary key*, pull_client_id yang merupakan id pengguna aplikasi *pull*, poster_id yang merupakan id poster yang dikirimkan, requested_at yang merupakan

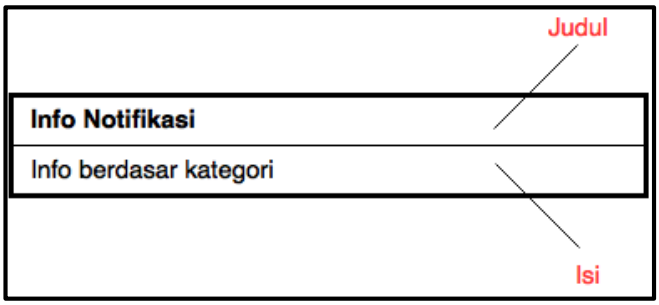
waktu data notifikasi diminta oleh aplikasi UMN Bulletin, got_at yang merupakan waktu data notifikasi diterima, dan viewed_at yang merupakan waktu notifikasi diklik oleh pengguna aplikasi *pull*. Tabel notification_pushed dan notification_pulled digunakan untuk mendapatkan hasil penelitian dengan *hands-on measurement*. Struktur tabel notification_pulled dijelaskan pada Tabel 3.22.

Tabel 3.22 Struktur Tabel Notification_Pulled

Nama Kolom	**Tipe Data**	**Panjang**
id (PK)	bigint	20
pull_client_id (FK)	bigint	20
poster_id (FK)	int	11
requested_at	datetime	-
got_at	datetime	-
viewed_at	datetime	-

3.2.5 Desain Antarmuka Notifikasi

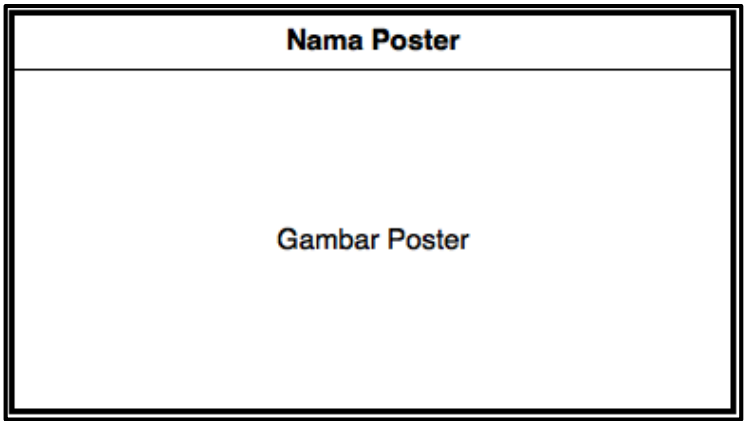
Gambar 3.29 merupakan Desain Antarmuka Notifikasi Normal. Terdapat dua elemen pada desain ini, yaitu judul dan isi. Judul dirancang untuk menampilkan kata "Informasi". Sedangkan isi dirancang untuk menampilkan kata-kata yang mengajak pengguna untuk melihat detil poster berdasarkan kategori poster. Misal, untuk kategori lomba, "Ada Lomba baru! Ayo daftar sekarang!".



Gambar 3.29 Desain Antarmuka Notifikasi Normal

Gambar 3.30 merupakan Desain Antarmuka Detil Notifikasi. Detil Notifikasi dirancang untuk menampilkan nama dan gambar dari poster. Untuk dapat menampilkan detil notifikasi, seperti yang telah dijelaskan pada bagian Activity

Diagram Melihat Detil Notifikasi, pengguna harus melakukan gerakan menyapu (*swipe*) notifikasi ke bawah dengan dua jari.



Gambar 3.30 Desain Antarmuka Detil Notifikasi