

## PERANCANGAN APLIKASI PUSH NOTIFICATION CENTER DENGAN TEKNOLOGI FIREBASE CLOUD MESSAGING DI PT. SUMBER TRIJAYA LESTARI

**Ramos Somya**

Fakultas Teknologi Informasi, Program Studi S1 Teknik Informatika  
Universitas Kristen Satya Wacana  
Email: ramos.somya@uksw.edu

**Monika Aprillia**

Fakultas Teknologi Informasi, Program Studi S1 Teknik Informatika  
Universitas Kristen Satya Wacana  
Email: 672015030@student.uksw.edu

### ABSTRAK

PT. Sumber Trijaya Lestari merupakan perusahaan yang bergerak di bidang digital dan telah mengembangkan berbagai macam usaha yang dituangkan ke dalam aplikasi di berbagai macam platform yaitu *web* dan *mobile*. *Push notification* disertakan dalam aplikasi yang dibangun supaya pengguna bisa mendapatkan informasi terkini berkaitan dengan penggunaan aplikasi-aplikasi tersebut. Kendala yang dihadapi saat ini adalah, setiap aplikasi memiliki masing-masing *dashboard* dan pengirim harus mencari token *client* target secara manual pada basis data, sehingga menyebabkan kinerja *developer* yang melakukan *push notification* menjadi tidak optimal. Hal ini disebabkan karena *push notification* hanya dapat dibuat oleh *developer* atau orang yang memahami Firebase, penggunaan Postman serta harus memiliki akses ke basis data. Oleh karena itu, pada penelitian ini dirancang sebuah aplikasi *push notification center* untuk mempermudah dalam pengiriman *push notification* dari setiap aplikasi melalui satu *dashboard*. Aplikasi ini dikembangkan berbasis *web* dengan memanfaatkan teknologi Java, JavaScript, basis data MySQL dan *Framework* Spring Boot. Berdasarkan hasil pengujian yang dilakukan kepada *developer* dan pengguna aplikasi, dapat disimpulkan bahwa sistem *push notification* berbasis *web* ini dapat berfungsi dengan baik serta dapat menyelesaikan permasalahan yang ada.

**Kata kunci:** *push notification; firebase messaging; spring boot.*

### ABSTRACT

*PT. Sumber Trijaya Lestari is one of the digital business companies who has been developing various applications in mobile and web platform. In order to provide the current information toward their use, those applications using push notification to send the information. However, there are problems in push notification delivery where each application has its own dashboard and the sender need to retrieve the client's token as the target manually from database which causes the performance of developer who do push notifications to be not optimal because it can only be done by developers or certain people who understand Firebase and the used of Postman and must have the access to database. Based on these problems, an aggregator is needed to facilitate push notification delivery for each application from a user friendly dashboard so it can be done by other divisions. Push Notification Center is a web-based application designed using the Java programming language, JavaScript, MySQL for database creation, Spring Boot. Based on the system testing from the developer side and the user side using the interview method, it can be concluded that the system can be used properly and be able to solve the problems.*

**Keywords:** *push notification; firebase messaging; spring boot.*

## 1. PENDAHULUAN

*Push notification* merupakan salah satu media penyampaian informasi dari *developer* suatu aplikasi kepada penggunanya baik pada aplikasi *web* maupun *mobile*. *Push notification* dapat mengirimkan informasi terbaru mengenai aplikasi tersebut secara *real time*, sehingga pesan yang penting dan relevan dapat tersampaikan kepada pengguna bahkan tanpa aplikasi dijalankan sekalipun [1].

Sebagai perusahaan yang bergerak di bidang *digital business*, PT. Sumber Trijaya Lestari telah mengembangkan berbagai macam usaha yang dituangkan ke dalam aplikasi di berbagai macam *platform* yaitu *web* dan *mobile*. Untuk memberikan informasi kepada pengguna aplikasi terlebih pada *platform*

*mobile*, perusahaan tentu menggunakan teknologi *push notification* yang dikelola dengan menggunakan *Firebase Cloud Messaging*, yaitu *Back-End as A Service* yang dikembangkan oleh Google. Namun banyaknya aplikasi yang telah dikembangkan menimbulkan permasalahan dalam pengelolaan dan pengiriman *push notification* karena setiap aplikasi memiliki *dashboard*nya masing-masing pada *Firebase console*. Pengiriman *push notification* ini hanya dapat dilakukan oleh *developer* atau orang-orang tertentu yang paham mengenai *Firebase* dan *Postman* serta memiliki akses *database*. Hal ini menyebabkan kegiatan *push notification* menjadi kurang efisien dan efektif dikarenakan harus seorang *developer* yang melakukan serta harus mencari *token user* secara manual pada *database* sehingga membutuhkan waktu untuk melakukannya yang menyebabkan kinerja *developer* menjadi tidak optimal. Hal ini juga berhubungan dengan masih kurangnya sumber daya manusia pada bagian *developer*.

Oleh karena itu dibutuhkan suatu media sebagai *aggregator* yang dapat mempermudah pengelolaan dan pengiriman *push notification* untuk berbagai macam aplikasi yang dikembangkan PT Sumber Trijaya Lestari dalam satu *dashboard* yang lebih *user friendly* yang memungkinkan *push notification* dilakukan oleh divisi lainnya. Berdasarkan latar belakang tersebut, maka rumusan masalah dalam penelitian ini adalah bagaimana merancang dan membangun sebuah aplikasi berbasis *web* sebagai *Push Notification Center* di PT Sumber Trijaya Lestari yang memanfaatkan layanan *Firebase Cloud Messaging* menggunakan *framework Spring Boot*.

*Firebase* merupakan sebuah *platform* untuk mendukung pengembangan aplikasi berbasis *mobile* dan *web* atau disebut juga sebagai *Backend as a Service (BaaS)*. Salah satu fitur dari *Firebase* adalah *Firebase Messaging Cloud (FCM)*, yaitu layanan yang digunakan untuk mengelola pengiriman *push notification* [2]. Sedangkan *Spring Boot* didesain untuk menyederhanakan pengembangan aplikasi yang menggunakan *framework Spring*. Beberapa konsep penting yang terdapat pada *Spring Boot* adalah konfigurasi aplikasi otomatis (*Automatic Configuration*), integrasi *dependency/library* yang dibutuhkan secara otomatis (*Starter Dependences*), memungkinkan kontrol aplikasi melalui *console (Command-Line-Interpreter)*, dan menampilkan informasi mengenai *event* yang ada di aplikasi (*The Actuator*) [3].

Penelitian yang berkaitan dengan pemanfaatan *Firebase Cloud Messaging* untuk mengelola *push notification* sering kali lebih fokus membahas dari sisi *client* atau penerima pesan, yaitu apakah *push notification* berhasil diterima oleh *client* setelah *FCM* diterapkan pada aplikasi yang telah dirancang. Pada penelitian yang berjudul Implementasi *Push Notification* pada Informasi Perkuliahan dan Kegiatan Mahasiswa Berbasis Android membahas mengenai permasalahan yang ada pada Ukrida (Universitas Kristen Krida Wacana), yaitu masih kurangnya publikasi mengenai informasi seputar perkuliahan maupun kegiatan-kegiatan yang diadakan oleh kampus. Untuk mengatasi permasalahan tersebut dibutuhkan suatu media yang dapat menyediakan informasi seputar perkuliahan serta kegiatan-kegiatan yang diadakan dengan memanfaatkan banyaknya pengguna *smartphone* dewasa ini [4].

Pada penelitian berjudul *Push Notification System* pada *Prototype* Kendali Listrik Rumah membahas mengenai permasalahan pemakaian listrik yang tidak efisien dikarenakan teknologi *timer* dianggap kurang memenuhi kebutuhan manusia dalam mengelola pemakaian listrik. Oleh karena itu dirancang sebuah sistem yang memanfaatkan *push notification* untuk memberikan peringatan untuk melakukan kendali dalam efisiensi penggunaan listrik [5].

Pada penelitian yang berjudul Perancangan dan Pembuatan Sistem Pengumuman Akademis Berbasis *Tag* Menggunakan *REST Web Service* membahas mengenai kurang efektifnya penyebaran pengumuman menggunakan media sosial. Untuk mengatasi hal tersebut dikembangkan sebuah sistem pengumuman khusus bersifat multi *platform* dengan menggunakan label penanda (*tag*) yang berguna untuk menyaring atau mengelompokkan pengumuman agar sesuai dengan kebutuhan penerima serta memanfaatkan *Firebase Cloud Messaging* untuk memberikan fitur *push notification* [6].

Pada penelitian yang berjudul *A Prototype Framework for High Performance Push Notification* membahas mengenai bagaimana sebuah *software prototype framework* dibangun untuk mengirimkan jutaan *push notification* dengan memanfaatkan *cloud service* dari *Google Cloud Messaging (kini Firebase Cloud Messaging)* dan *APNS (Apple Push Notification Service)* [1].

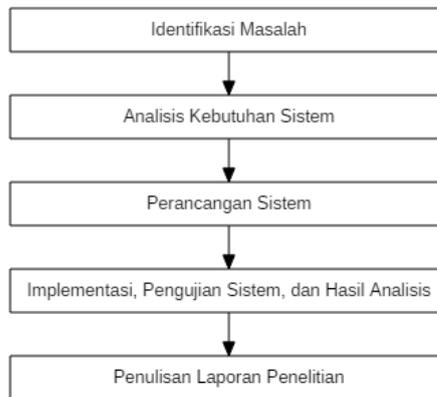
Pada penelitian yang berjudul Pengembangan Sistem Informasi 3 Pilar dalam Penyelesaian Perkara Tilang di Kota Kediri Menggunakan *RESTful Web Service* membahas mengenai permasalahan mekanisme penyelesaian perkara tilang di Kota Kediri di antaranya, pengaturan data yang tidak terorganisir, pembuatan laporan yang memakan waktu, kurangnya informasi yang diterima masyarakat, dan sulitnya akses kepada instansi pemerintahan lain di Kota Kediri. Untuk menyelesaikan permasalahan ini dibuatlah sistem informasi menggunakan *RESTful web service* dengan *framework Spring Boot* [7].

Berdasarkan beberapa penelitian yang telah membahas mengenai pemanfaatan *Firebase Cloud Messaging (FCM)* dan *framework Spring Boot* maka disusunlah penelitian terkait penggunaan *FCM* untuk mengelola *push notification* serta implementasi *framework Spring Boot*. Berbeda dari penelitian sebelumnya, pada penelitian ini akan dibangun sebuah aplikasi berbasis *web* yang menggunakan *framework Spring Boot* sebagai *Push Notification Center* berupa *dashboard* yang lebih *user friendly*

untuk pengiriman *push notification* untuk aplikasi *mobile* (Android dan IOS) di mana untuk pengelolaannya menggunakan layanan *FCM* dan difokuskan dari sisi *admin* atau pengirim.

## 2. METODOLOGI PENELITIAN

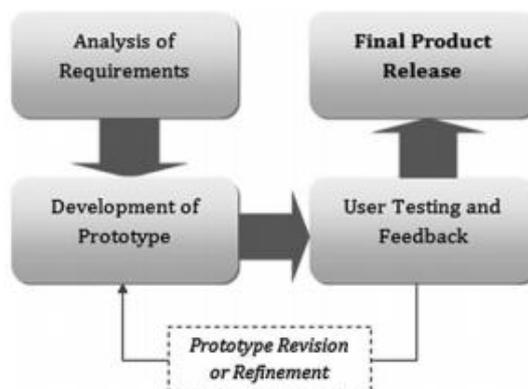
Penelitian ini dilakukan dan diselesaikan melalui lima tahapan penelitian yang dapat dilihat pada Gambar 1.



**Gambar 1. Tahapan Penelitian**

Berikut penjelasan tahapan penelitian berdasarkan Gambar 1: 1) Pada tahap identifikasi masalah dilakukan identifikasi dan analisis terhadap masalah yang ada, yaitu bagaimana membangun *Push Notification Center* sehingga dapat mengaktifkan kegiatan *push notification* pada PT Sumber Trijaya Lestari. 2) Pada tahap analisis kebutuhan sistem dilakukan wawancara terhadap karyawan perusahaan untuk mendapatkan informasi mengenai kebutuhan sistem yang akan dibangun. 3) Pada tahap perancangan sistem dilakukan perancangan keseluruhan sistem, meliputi perancangan *Unified Modeling Language (UML)*, perancangan *database* dan perancangan antarmuka di mana perancangan sistem dilakukan menggunakan metode *prototype*. 4) Pada tahap implementasi, pengujian sistem, dan hasil analisis dilakukan implementasi dari perancangan sistem di tahap sebelumnya dan kemudian diuji untuk memastikan setiap fungsi pada sistem berjalan dengan baik dan sesuai dengan fungsinya. Kemudian dilakukan penyimpulan dari hasil penelitian yang telah dilakukan. 5) Pada tahap penulisan laporan penelitian akan dilakukan dokumentasi terhadap proses penelitian dari awal hingga selesai dalam bentuk artikel ilmiah.

Metode yang digunakan dalam perancangan sistem adalah metode *prototype* karena keterlibatan *user* yang tinggi dalam memantau pengembangan aplikasi memungkinkan hasil akhir aplikasi dapat sesuai dengan permintaan *user*. Metode *prototype* ini mengizinkan *user* untuk melihat pengembangan aplikasi secara langsung sehingga dapat langsung memberikan *feedback* kepada *developer* ketika ada spesifikasi yang kurang tepat bahkan juga menambahkan permintaan spesifikasi sistem [8].

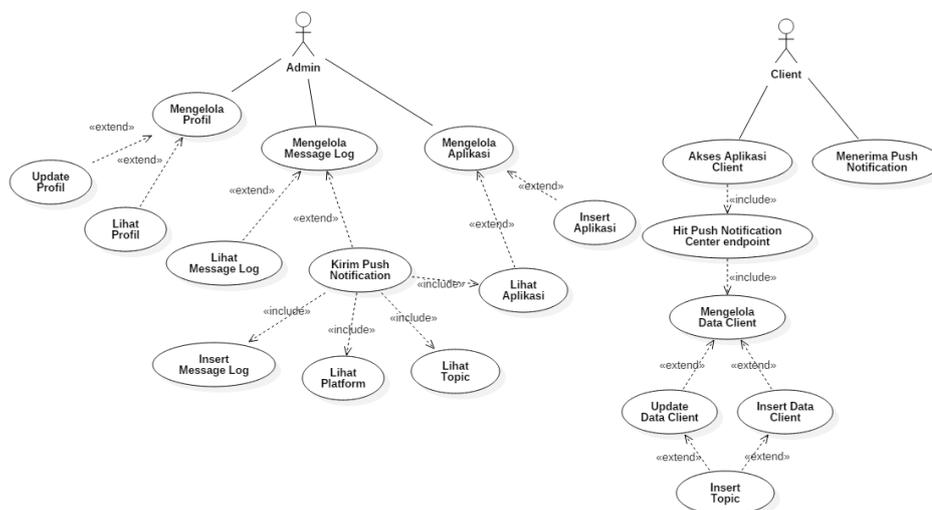


**Gambar 2. Metode *Prototype* [8]**

Tahap pertama yaitu *Analysis of Requirements* yang merupakan proses identifikasi dan analisis kebutuhan *user*. Hasil identifikasi dan analisis dari penelitian ini adalah dibutuhkan suatu sistem yang *user*

*friendly* yang dapat membantu pengiriman *push notification* ke berbagai macam aplikasi *client* pada *platform mobile* hanya melalui satu *dashboard*. Tahap kedua yaitu *Development of Prototype* yang merupakan tahap pengembangan aplikasi, pada tahap ini peneliti melakukan perancangan sistem dengan menggunakan *Unified Modeling Language (UML)*, perancangan *database*, serta perancangan *interface* sistem sesuai dengan permintaan *user* dan kemudian dilakukan pengembangan sistem berdasarkan rancangan yang telah dibuat. Tahap ketiga merupakan *User Testing and Feedback* yang merupakan proses pengujian sistem dan evaluasi sistem dari *user*. Pada tahap ini peneliti melakukan pengujian terhadap aplikasi yang telah dikembangkan untuk memastikan bahwa sistem sesuai dengan kebutuhan *user*, kemudian *user* akan memberikan *feedback* ketika dibutuhkan perbaikan atau ketika ada permintaan tambahan dari *user*. Ketika dibutuhkan perbaikan maka akan kembali pada tahap 2. Peneliti akan melakukan pengembangan aplikasi untuk melakukan perbaikan dan akan kembali dilakukan tahap 3 yaitu tahap pengujian. Proses ini akan terus berlangsung ketika sistem belum sesuai dengan permintaan *user*. Ketika sudah sesuai kemudian akan dilanjutkan ke tahap 4 yaitu *Final Product Release* yang merupakan hasil akhir dari aplikasi yang dikembangkan ketika sudah sesuai dengan kebutuhan *user*.

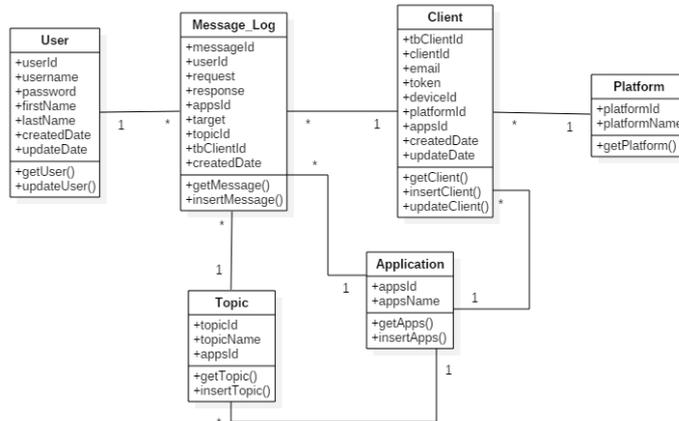
Perancangan sistem dibuat menggunakan *Unified Modeling Language (UML)*, yang merupakan standar yang digunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak berupa diagram [9] [10]. Adapun *UML* dari *Push Notification Center* yang akan dijelaskan meliputi *use case diagram*, *activity diagram*, *sequence diagram* dan *class diagram*.



**Gambar 3. Use Case Diagram Push Notification Center**

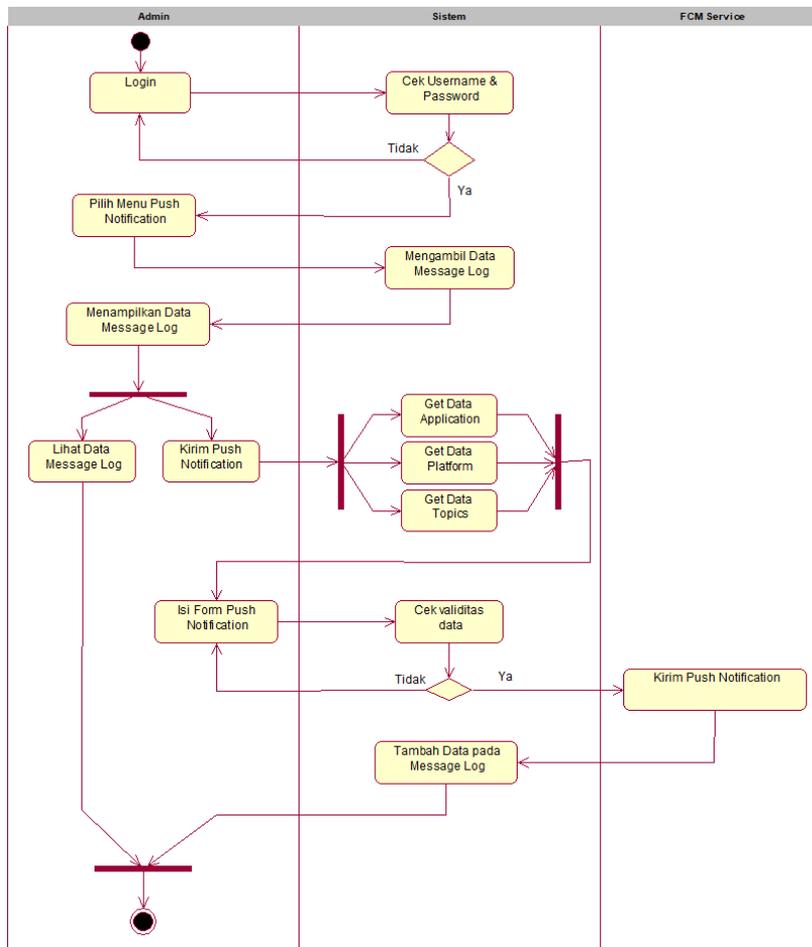
*Use case diagram* berfungsi juga untuk menggambarkan kebutuhan pengguna sistem [11]. Berdasarkan Gambar 3 terdapat dua pengguna pada *Push Notification Center* yaitu *admin* web dan *client* aplikasi. *Admin* memiliki 3 fungsi utama yaitu mengelola *Message Log*, mengelola Aplikasi, dan mengelola Profil. Pada *case* mengelola *message log* terdapat 2 fungsi yaitu, lihat data *message log* dan kirim *push notification*. Pada fungsi kirim *push notification* *Admin* dapat melihat daftar aplikasi, *platform*, serta *topic* yang terdaftar pada web saat pengisian *form* untuk *push notification* dan ketika *push notification* berhasil secara otomatis akan *insert* pada data *message log*. Pada *case* mengelola aplikasi, *Admin* dapat melihat daftar aplikasi yang terdaftar pada sistem (saat pengisian *form* kirim *push notification*) dan mendaftarkan aplikasi baru pada sistem setelah mendaftarkannya pada *Firestore* terlebih dahulu. Pada *case* mengelola profil, *Admin* dapat melihat data profilnya dan memperbaruinya.

*Client* memiliki 2 fungsi utama yaitu Akses Aplikasi *Client* dan Menerima *Push Notification*. Pada *case* akses aplikasi *client* ketika seorang *client* mengakses aplikasi maka secara otomatis akan menembak *endpoint* dari web *Push Notification Center*. Ketika *endpoint* tersebut diakses maka secara otomatis akan menambahkan data baru *client* (*insert data client*) jika belum terdaftar atau memperbarui data (*update data client*) jika sudah terdaftar dan juga akan menambahkan nama *topic* (*insert topic*) jika *topic* pada aplikasi tersebut belum terdaftar. Pada *case* menerima *push notification* *client* akan menerima *push notification* yang dikirimkan oleh *Admin*.



**Gambar 4. Class Diagram Push Notification Center**

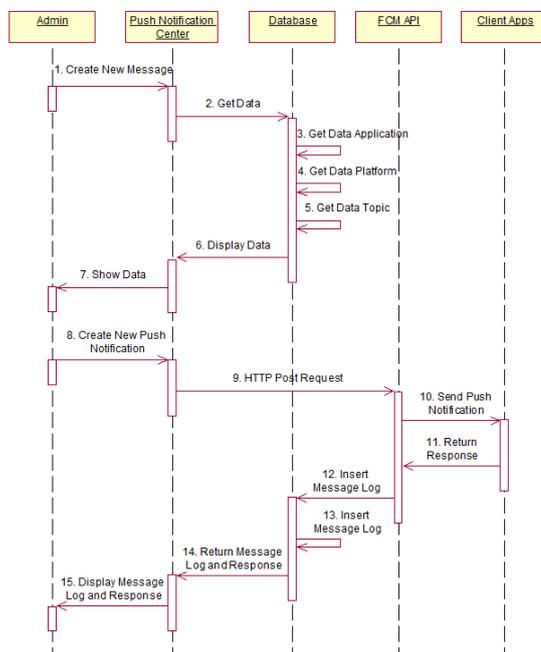
*Class Diagram* dapat digunakan untuk menggambarkan konseptual skema dari sebuah sistem [12]. Berdasarkan *class diagram* pada Gambar 4, *Push Notification Center* memiliki 6 *class* yaitu *User*, *Message\_Log*, *Client*, *Platform*, *Topic*, dan *Application*. Setiap *class* memiliki *attribute* dan *operation* masing-masing serta memiliki relasi satu dengan lainnya sesuai hubungan antar *class*.



**Gambar 5. Activity Diagram Mengelola Message Log**

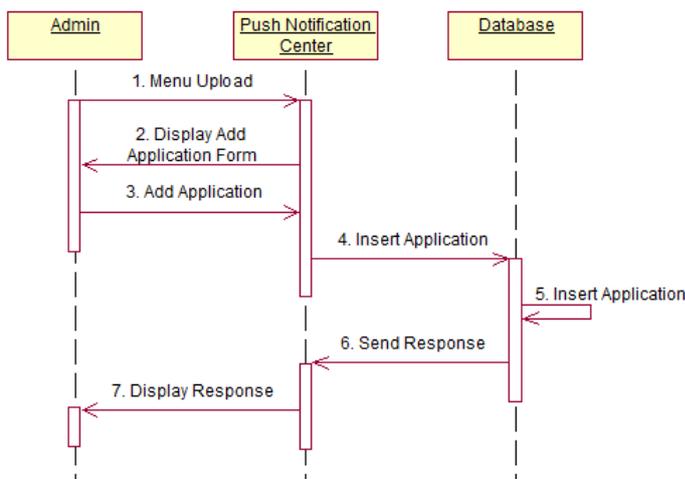
Gambar 5 menunjukkan *activity diagram* dari Mengelola *Message Log* yaitu lihat data *message log* dan kirim *push notification*. Pertama seorang *Admin* harus melakukan *login* dan setelah berhasil dapat memilih menu *Push Notification*. Kemudian sistem akan mengambil data *message log* dari *database* dan menampilkannya pada halaman menu *Push Notification*. Pada menu tersebut *Admin* dapat melakukan dua aktivitas yaitu hanya melihat data *message log* atau mengirimkan *push notification*. Ketika *Admin*

memilih menu kirim *Push Notification* maka sistem akan menampilkan *form* untuk data *push notification* termasuk data aplikasi, data *platform* dan data *topic*. Kemudian data akan dicek berdasarkan ketentuan-ketentuan yang ada ketika sudah memenuhi ketentuan, *push notification* akan *request* pada *API Firebase Cloud Messaging (FCM)* yang kemudian akan mengirimkannya pada *Client*. Kemudian data *push notification* beserta respon dari *FCM* tersebut akan ditambahkan pada *message log*.



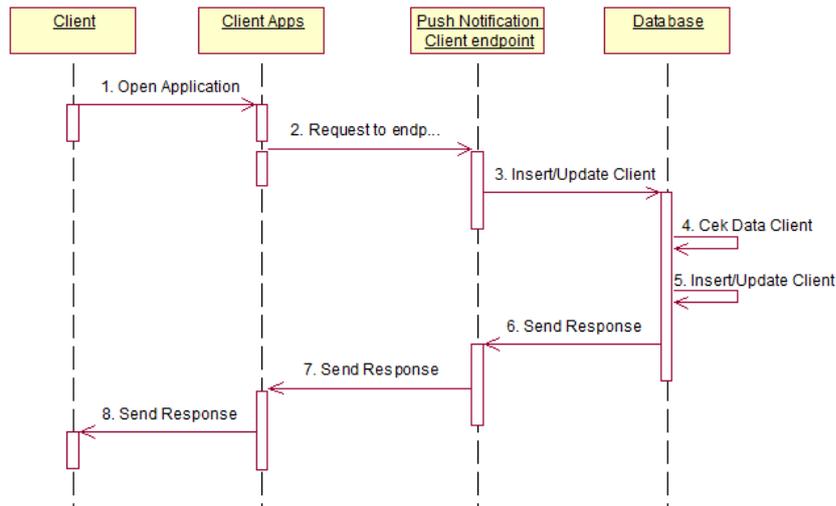
**Gambar 6. Sequence Diagram Send Push Notification**

Gambar 6 menunjukkan *sequence diagram* pengiriman *push notification*. Proses dimulai dengan *Admin* mengakses menu *Push Notification* pada *Push Notification Center* kemudian ketika menampilkan *form* sistem juga akan menampilkan data *application*, *platform*, dan *topic* pada *form* yang diambil dari *database*. Setelah *Admin* mengirimkan *push notification* dari *Dashboard* maka sistem akan *request* pada *FCM service* untuk mengirimkan pada *client*. Setelah terkirim maka akan mengembalikan response berupa sukses atau gagal. Data *push notification* tersebut kemudian akan ditambahkan pada *message log* dan ditampilkan kembali pada tabel *message log*.



**Gambar 7. Sequence Diagram Add Application**

Gambar 7 menunjukkan *sequence diagram* pendaftaran aplikasi pada sistem. Pertama *Admin* mengakses menu *Upload* dan sistem akan menampilkan *form* untuk *insert* aplikasi. Kemudian *Admin* dapat mengisi *form* sesuai ketentuan dan kemudian tambahkan dan sistem akan menyimpannya pada *database*.

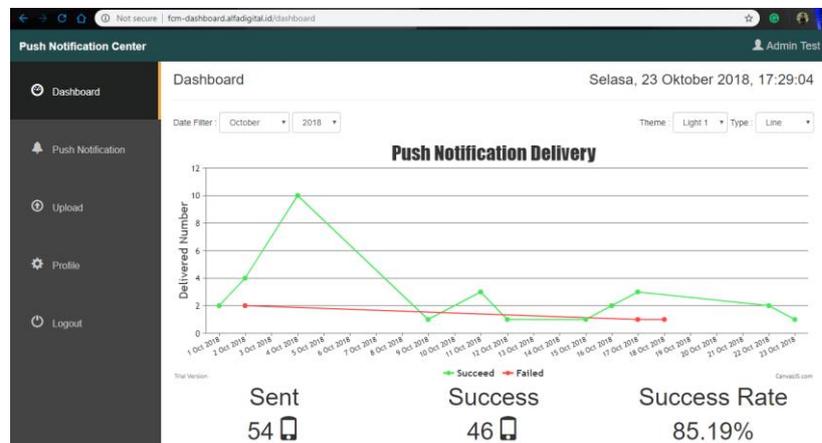


Gambar 8. Sequence Diagram Insert/Update Client

Pada Gambar 8 menunjukkan *sequence diagram* dari proses *insert* dan *update* data *client*. Ketika *client* mengakses aplikasi *client*, aplikasi tersebut akan menembak *endpoint* dari web *Push Notification Center* secara otomatis dan mengirimkan data *client*. Kemudian sistem melakukan pengecekan data *client* pada *database*, ketika *client* telah terdaftar maka data tersebut akan diperbarui tetapi jika *client* belum terdaftar maka akan ditambahkan data baru. Kemudian sistem akan mengembalikan respons apakah data berhasil diupdate/insert.

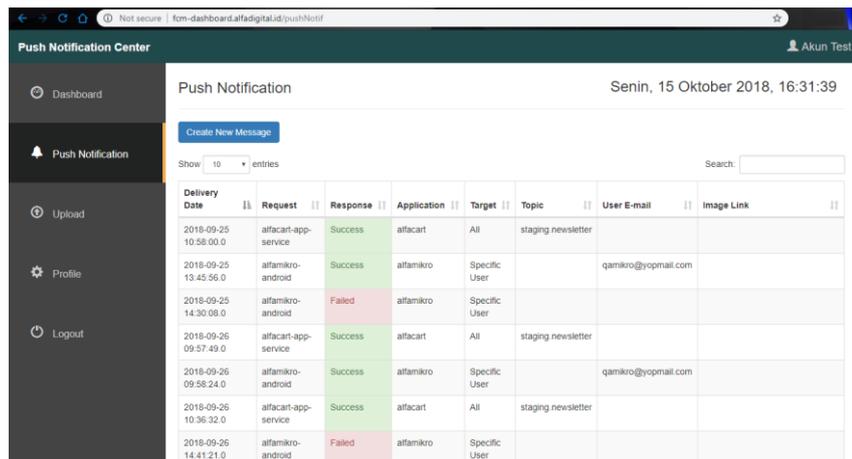
### 3. HASIL DAN PEMBAHASAN

Berdasarkan perancangan sistem yang ada maka dilakukan implementasi dan pengujian sistem *Push Notification Center*. Implementasi dan pengujian sistem ini diharapkan dapat sesuai kebutuhan pengguna sehingga dapat menyelesaikan permasalahan yang ada.



Gambar 9. Menu Dashboard Push Notification Center

Gambar 9 menunjukkan tampilan menu *Dashboard Push Notification Center*. Halaman ini menyajikan informasi mengenai total pengiriman *push notification* dalam bentuk grafik yang dibedakan berdasarkan responnya serta beberapa keterangan seperti total pengiriman *push notification* (*Sent*), total *push notification* yang sukses terkirim (*Success*) dan persentase *push notification* yang sukses dari total pengiriman. Pada halaman ini terdapat fungsi *filtering* untuk memilih waktu dari data yang ditampilkan pada grafik yaitu berdasarkan bulan dan tahun. Adapun fungsi tambahan lainnya yaitu *Admin* dapat memilih tema tampilan dan jenis grafik yang ditampilkan.

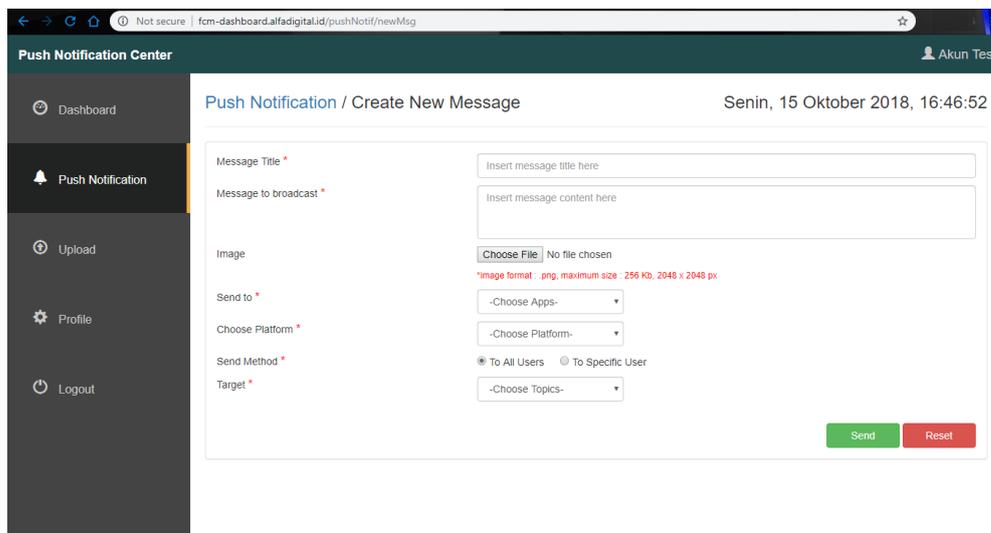


The screenshot shows the 'Push Notification Center' dashboard. On the left is a sidebar menu with options: Dashboard, Push Notification, Upload, Profile, and Logout. The main content area is titled 'Push Notification' and shows a 'Create New Message' button. Below it is a table with columns: Delivery Date, Request, Response, Application, Target, Topic, User E-mail, and Image Link. The table contains 10 rows of data with 'Success' and 'Failed' status indicators.

Delivery Date	Request	Response	Application	Target	Topic	User E-mail	Image Link
2018-09-25 10:58:00.0	alfacart-app-service	Success	alfacart	All	staging newsletter		
2018-09-25 13:45:56.0	alfamiko-android	Success	alfamiko	Specific User		qamiko@yopmail.com	
2018-09-25 14:30:00.0	alfamiko-android	Failed	alfamiko	Specific User			
2018-09-26 09:57:49.0	alfacart-app-service	Success	alfacart	All	staging newsletter		
2018-09-26 09:58:24.0	alfamiko-android	Success	alfamiko	Specific User		qamiko@yopmail.com	
2018-09-26 10:36:32.0	alfacart-app-service	Success	alfacart	All	staging newsletter		
2018-09-26 14:41:21.0	alfamiko-android	Failed	alfamiko	Specific User			

Gambar 10. Menu *Push Notification Center*

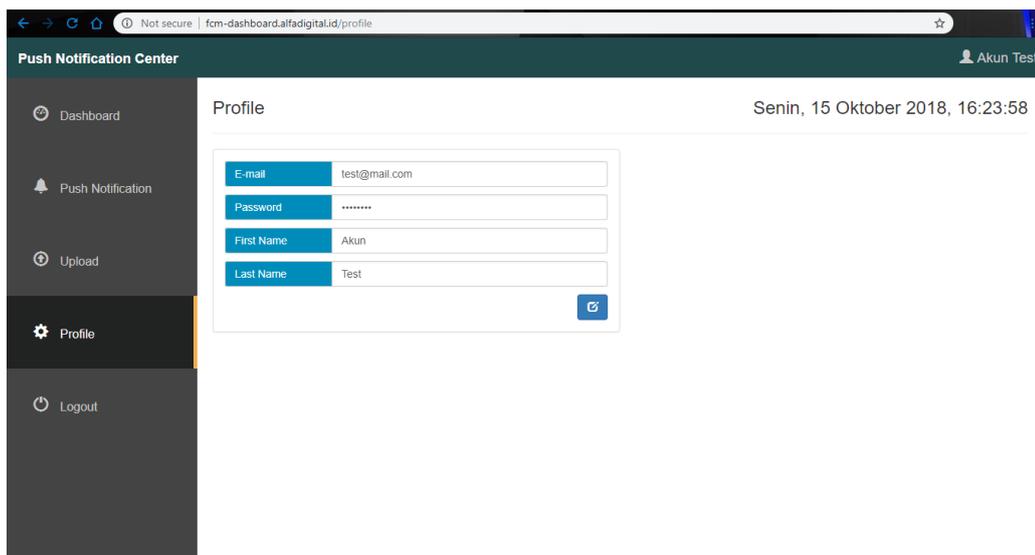
Gambar 10 menunjukkan tampilan halaman *Push Notification*. Halaman ini menampilkan data *message log* dari pengiriman *push notification*. Terdapat 2 fungsi tambahan pada tabel yaitu *search* dan *sorting* data di setiap kolomnya. Pada halaman ini *Admin* juga dapat membuat *push notification* baru pada menu *Create New Message*.



The screenshot shows the 'Push Notification / Create New Message' form. It includes fields for 'Message Title', 'Message to broadcast', 'Image' (with a 'Choose File' button and a note: '\*Image format : png, maximum size : 256 Kb, 2048 x 2048 px'), 'Send to' (dropdown), 'Choose Platform' (dropdown), 'Send Method' (radio buttons for 'To All Users' and 'To Specific User'), and 'Target' (dropdown). There are 'Send' and 'Reset' buttons at the bottom right.

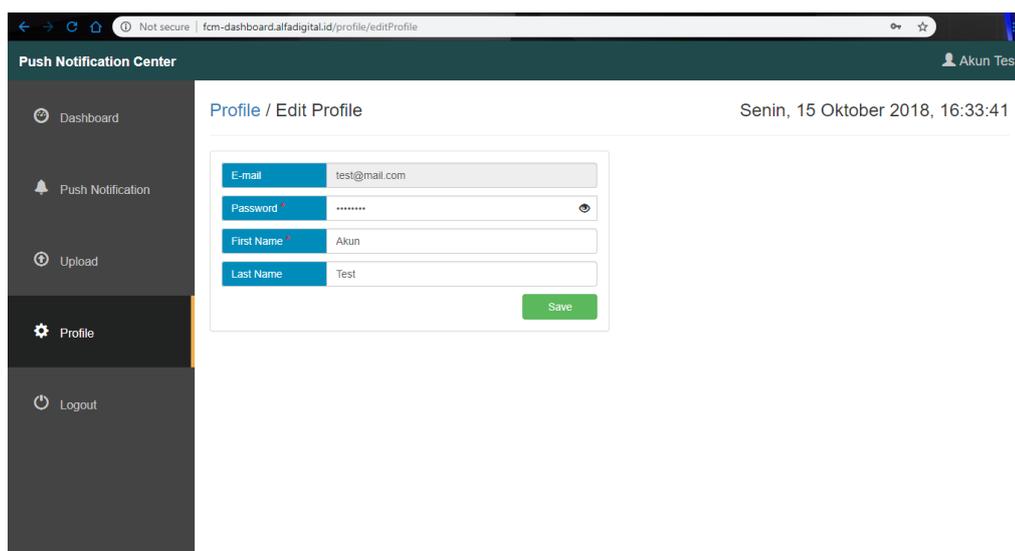
Gambar 11. Menu *Create New Message* pada *Push Notification Center*

Gambar 11 menunjukkan tampilan menu *Create New Message* dari halaman *Push Notification*. Untuk mengirimkan *push notification* *Admin* dapat mengisi *form* yang tersedia pada halaman tersebut kemudian mengirimnya. Untuk *field Target* akan berubah berdasarkan pilihan *Send Method*. Ketika memilih *To All Users* pada *field Target* akan menampilkan daftar nama *Topic* berdasarkan aplikasi yang dipilih dalam bentuk *dropdown* dan ketika memilih *To Specific User* pada *field Target* akan menampilkan *textfield* untuk memasukkan email *client* yang menjadi target.



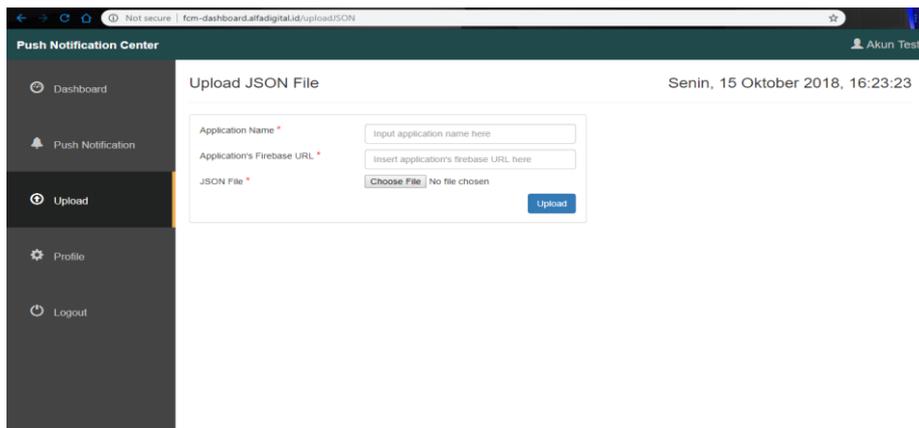
**Gambar 12. Menu Profile Push Notification Center**

Gambar 12 menunjukkan tampilan menu *Profile*. Pada halaman tersebut memberikan informasi mengenai data profil *Admin* seperti *email*, *password*, *first name*, dan *last name*. *Admin* juga dapat memperbarui data tersebut.



**Gambar 13. Menu Edit Profile Push Notification Center**

Gambar 13 menunjukkan halaman *Edit Profile* dari menu *Profile*. Pada halaman ini *Admin* dapat memperbarui data profilnya yaitu *password*, *first name*, dan *last name* sedangkan *email* tidak dapat diubah berdasarkan permintaan *user*.



**Gambar 14. Menu Upload Push Notification Center**

Gambar 14 menunjukkan tampilan menu *Upload*. Pada menu ini *Admin* dapat menambahkan atau mendaftarkan aplikasi baru pada sistem setelah mendaftarkannya pada *Firebase* terlebih dahulu. Dari *Firebase* tersebut akan diperoleh *JSON file* aplikasi tersebut yang kemudian diunggah bersamaan ketika menambahkan aplikasi baru.

```
GoogleCredential googleCredential = GoogleCredential.  
    fromStream(new FileInputStream(environment.  
        getProperty("fcm-key.json.path") + file)).createScoped(Arrays.  
        asList("https://www.googleapis.com/auth/firebase.messaging"));  
googleCredential.refreshToken();  
return googleCredential.getAccessToken();
```

Terdapat beberapa metode untuk melakukan *push notification* dengan menggunakan *FCM service* salah satunya adalah *HTTP Post*. Pada metode ini dilakukan *request* terhadap *FCM API* yang terdiri dari *header* dan *body* dengan data berformat *JSON*. *Header* dari *HTTP Post request* ini berisi *Content-Type* yang menunjukkan tipe atau format dari data yang dikirim dan *Authorization* berupa *token* yang diperoleh dengan menggunakan *Google Credential*. Untuk menghasilkan *token* ini *Google Credential* membutuhkan *JSON file* dari aplikasi target *push notification* yang diperoleh ketika mendaftarkan aplikasi pada *firebase console* dan kemudian mengirimkannya pada *Firebase Messaging API* yang kemudian akan mengembalikan *response* berupa *token*.

Kode program di atas merupakan fungsi *getAccessToken* untuk menghasilkan *token Authorization*. Pada fungsi ini *Google Credential* akan membaca *JSON file* aplikasi yang telah disimpan pada *server* dimana *path* penyimpanan *file* dideklarasikan pada *application.properties* dengan nama "*fcm-key.json.path*".

```
RestTemplate restTemplate = new RestTemplate();  
Application application = appsDAO.getDataById(app);  
String apiURL = environment.getProperty("firebase-url.post") +  
    application.getFbURL() + "/messages:send";  
String jsonFile = application.getJsonFile();  
HttpHeaders headers = new HttpHeaders();  
headers.set("Content-Type", "application/json");  
headers.set("Authorization", "Bearer " + getAccessToken(jsonFile));
```

Kode program di atas merupakan fungsi *send* dengan tipe pengembalian berupa kode status dari *response FCM API* untuk mengirimkan *push notification*. Pada fungsi ini menggunakan *Rest Template* untuk melakukan *HTTP Post*. Baris 2-5 berfungsi untuk mengakses data aplikasi dari *database* karena membutuhkan nama *JSON file* dan *firebase URL* dari aplikasi. Kemudian menambahkan *header*, yaitu *Content-Type* dan *Authorization* menggunakan *HttpHeader*. Kemudian *header* dan *body* disatukan menggunakan *HttpEntity* yang kemudian akan dikirimkan ke *FCM API* menggunakan *Rest Template*. Pengiriman tersebut dideklarasikan pada *Response Entity* untuk memperoleh kode status dari *response*.

```
data.put("title", dataTitle);  
data.put("body", dataBody);  
data.put("image", imgLink);  
return data;
```

Karena format data dari *body* adalah *JSON* maka dibutuhkan fungsi dengan tipe data *JSON*. Kode program di atas menunjukkan potongan fungsi untuk mengirimkan konten *data* untuk *platform android*.

```

alert.put("body", dataBody);
alert.put("title", dataTitle);
aps.put("alert", alert);
aps.put("badge", 1);
aps.put("sound", "default");
aps.put("category", "CustomSampluPush");
aps.put("mutable-content", 1);
payload.put("aps", aps);
payload.put("urlImageString", imgLink);
headers.put("apns-priority", "10");
apns.put("headers", headers);
apns.put("payload", payload);
return apns;

```

Karena konten pada *body* dibedakan berdasarkan *platform* maka fungsi untuk menampung *body* untuk *platform android* dan *iOS* dibedakan. Kode program di atas menunjukkan potongan fungsi untuk memasukkan *konten apns* untuk *platform ios*.

Pengujian sistem dilakukan untuk memastikan sistem dapat berjalan dengan baik dan sesuai dengan kebutuhan pengguna. Terdapat 2 proses pengujian yang dilakukan peneliti, yaitu pengujian dari sisi peneliti dan dari sisi pengguna. Pengujian dari sisi *developer* atau peneliti dilakukan dengan metode *Black Box Testing* untuk memastikan setiap fungsi yang ada pada sistem berjalan dengan baik dan sesuai kebutuhan [13]. Pada pengujian ini peneliti mendefinisikan sejumlah kondisi *input* pada sistem, kemudian mengamati *output*, dan menentukan validitasnya berdasarkan kesesuaian *output* yang diharapkan dengan *output* yang terjadi. Hasilnya ada di Tabel 1.

**Tabel 1. Hasil black box testing**

<i>Fungsi</i>	<i>Skenario</i>	<i>Hasil</i>	<i>Validitas</i>
<i>Push Notification</i>	Kirim <i>push notification target</i> "All Users"	<i>Client</i> yang terdaftar pada topik pilihan menerima notifikasi	✓
<i>Push Notification</i>	Kirim <i>push notification target</i> "Specific User", <i>user</i> sudah terdaftar	<i>Client</i> tertentu menerima notifikasi	✓
<i>Push Notification</i>	Kirim <i>push notification</i> yang menyertakan gambar	Notifikasi terkirim dan menampilkan gambar	✓
<i>Message Log + Grafik</i>	Kirim <i>push notification</i>	Data pada <i>message log</i> dan grafik <i>terupdate</i> secara <i>real-time</i>	✓
<i>Add Application + upload JSON file</i>	Daftar aplikasi dan tambah <i>JSON file</i> pada sistem setelah mendaftarkan pada <i>firebase</i> terlebih dahulu	Aplikasi terdaftar, nama aplikasi ditampilkan pada daftar aplikasi tujuan <i>push notification</i>	✓
<i>Update Profile</i>	Edit data profil Admin	Data pada <i>database</i> dan halaman Profile <i>terupdate</i>	✓
<i>Client Registration</i>	<i>Client</i> baru pertama kali membuka aplikasi (belum melakukan proses <i>login</i> )	Aplikasi mengirimkan token <i>user</i> , <i>id device</i> , jenis <i>platform</i> , nama aplikasi, <i>topic</i> dari aplikasi tersebut dan tanggal terdaftar ke <i>endpoint</i> sistem.	✓
<i>Client Update</i>	<i>Client login</i> pada aplikasi	<i>Update</i> data menambahkan <i>id client</i> dan email	✓

Pengujian yang kedua merupakan *User Acceptance Testing* yaitu pengujian dari sisi pengguna yang dilakukan dengan metode wawancara. Pada tanggal 3 Oktober 2018 peneliti melakukan presentasi terhadap 3 orang dari divisi *Mobile API Specialist* berdasarkan sistem yang telah dibuat. Kemudian peneliti meminta tanggapan berupa kritik dan saran untuk pengembangan selanjutnya. Berdasarkan tanggapan yang diberikan dapat disimpulkan bahwa sistem dapat berfungsi dengan baik dan dapat membantu pengelolaan *push notification*. Ada beberapa penambahan fungsi yang telah diimplementasi dari desain awal sistem, yaitu penambahan menu *Upload* untuk mengunggah *JSON file* ketika ada

aplikasi baru yang akan didaftarkan pada sistem. Terdapat beberapa saran yang dapat diterapkan jika terdapat pengembangan sistem selanjutnya, yaitu penambahan fitur *delayed push notification*, di mana pesan dapat terkirim berdasarkan tanggal dan jam yang dipilih, penambahan informasi mengenai berapa banyak *client* mengakses aplikasi *client* dalam rentang waktu tertentu, dan pengembangan metode yang digunakan untuk mengirim *push notification* dengan menggunakan *admin SDK*.

#### 4. KESIMPULAN

Berdasarkan hasil penelitian dan pengujian dapat disimpulkan bahwa implementasi *web Push Notification Center* yang memanfaatkan *FCM* dan dibangun menggunakan *framework Spring Boot* dapat membantu pengelolaan dan pengiriman *push notification* pada PT. Sumber Trijaya Lestari. Dengan adanya sebuah media yang menjadi *aggregator* untuk setiap aplikasi yang dikembangkan oleh PT. Sumber Trijaya Lestari, pengiriman *push notification* dapat dilakukan dengan mudah melalui satu *dashboard* hanya dengan memilih aplikasi dan tujuan pengiriman. Sedangkan untuk pengiriman dengan target *client* tertentu pengirim tidak lagi harus mengakses *database* secara manual. Cukup dengan memasukkan email dari *client* sasaran dan sistem yang melakukan pencarian data pada *database* untuk memperoleh *token*. Adanya *Rest API* untuk pendaftaran *client* secara otomatis juga membantu dalam penambahan data *client* aplikasi sebagai sasaran *push notification* pada sistem.

Saran jika terdapat pengembangan selanjutnya, dapat dilakukan penambahan fitur *delayed push notification* sehingga *push notification* dapat terkirim berdasarkan tanggal dan jam yang dipilih. Dapat juga ditambahkan informasi mengenai berapa banyak *client* yang mengakses aplikasi *client* dalam rentang waktu tertentu. Selain itu dengan adanya berbagai macam metode yang dapat digunakan dalam pengiriman *push notification* menggunakan *Firestore Cloud Messaging*, diharapkan dapat dikembangkan dengan menggunakan metode lainnya yaitu *admin SDK* untuk lebih mengoptimalkan sistem dalam pengiriman *push notification*.

#### DAFTAR PUSTAKA

- [1] E. I. S. S. D. Kiliç, "A Prototype Framework for High Performance Push Notifications," *Int. J. Comput. Appl.*, vol. 166, no. 10, pp. 8–11, 2017.
- [2] D. K. T. Neha Srivastava, Uma Shree, Nupa Ram Chauhan, "FIREBASE CLOUD MESSAGING (ANDROID)," *Int. J. Innov. Res. Sci. Eng. Technol.*, vol. 6, no. 9, pp. 11–18, 2017.
- [3] S. R. Željko Jovanović, Dijana Jagodić, Dejan Vujičić, "JAVA SPRING BOOT REST WEB SERVICE INTEGRATION WITH JAVA ARTIFICIAL INTELLIGENCE WEKA FRAMEWORK," in *International Scientific Conference "UNITECH 2017,"* 2017, pp. 323–327.
- [4] F. Jefferson Setiawan, Edy Kristianto, "IMPLEMENTASI PUSH NOTIFICATION PADA INFORMASI PERKULIAHAN DAN KEGIATAN MAHAWSIWA BERBARIS ANDROID," *J. Tek. dan Ilmu Komput.*, vol. 4, no. 14, pp. 211–219, 2015.
- [5] D. E. Kurniawan, "PUSH NOTIFICATION SYSTEM PADA PROTOTYPE KENDALI LISTRIK RUMAH," *J. Comput. Eng. Syst. Sci.*, vol. 2, no. 1, pp. 89–92, 2017.
- [6] E. D. S. Santoso, "Perancangan dan Pembuatan Sistem Pengumuman Akademis Berbasis Tag Menggunakan REST Web Service," *J. Sist. Inf.*, vol. 8, no. 1, pp. 48–53, 2018.
- [7] P. A. N. A. H. B. A. P. Kharisma, "Pengembangan Sistem Informasi 3 Pilar Dalam Penyelesaian Perkara Tilang Di Kota Kediri Menggunakan Restful Web Service," *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 10, pp. 3525–3532, 2018.
- [8] T. Isaias, Pedro, Issa, *High level Models and Methodologies for Information Systems*. London: Springer, 2015.
- [9] A. Hendini, "PEMODELAN UML SISTEM INFORMASI MONITORING PENJUALAN DAN STOK BARANG (STUDI KASUS: DISTRO ZHEZHA PONTIANAK)," *J. KHATULISTIWA Inform.*, vol. 4, no. 2, pp. 107–116, 2016.
- [10] N. L. H. Yasir Dawood Salman, "Test Case Generation Model for UML Diagrams," *J. Telecommun. Electron. Comput. Eng.*, vol. 9, no. 2, pp. 171–175, 2017.
- [11] S. C. and C. C. Belén Bonilla-Morales, "Reuse of Use Cases Diagrams: An Approach based on Ontologies and Semantic Web Technologies," *Int. J. Comput. Sci. Issues*, vol. 9, no. 2, pp. 24–29, 2012.
- [12] A. Al-Shamailh, "An Experimental Comparison of ER and UML Class Diagrams," vol. 8, no. 2, pp. 279–288, 2015.
- [13] H. R. M. Sidi Mustaqbal, Roeri Fajri Firdaus, "PENGUJIAN APLIKASI MENGGUNAKAN BLACK BOX TESTING BOUNDARY VALUE ANALYSIS (Studi Kasus : Aplikasi Prediksi Kelulusan SNMPTN)," *J. Ilm. Teknol. Inf. Terap.*, vol. 1, no. 3, pp. 31–36, 2015.