BAB 6 FIREBASE CLOUD MESSAGING (FCM)

6.1 Latar Belakang

Firebase Cloud Messaging (FCM) adalah solusi pengiriman pesan lintas platform yang memungkinkan Anda mengirimkan pesan dengan tepercaya tanpa biaya. Dengan menggunakan FCM, Anda bisa mengirim pesan secara masif pada suatu kelompok atau individu. Selain itu dengan FCM anda bisa mengirim data pada aplikasi mobile yang sudah ditentukan. Untuk kasus penggunaan pengiriman pesan instan, pesan dapat mentransfer payload hingga 4 KB ke aplikasi klien.

6.1.1 Kemampuan Firebase Cloud Messaging

Mengirim pesan notification atau pesan data	Mengirim pesan notification yang ditampilkan kepada pengguna. Atau mengirim pesan data dan menentukan sepenuhnya apa yang terjadi dalam kode aplikasi
Penargetan pesan serbaguna	Mendistribusikan pesan ke aplikasi klien dengan salah satu dari tiga cara : ke satu perangkat, ke grup perangkat, atau ke perangkat yang berlangganan topik.
Mengirim pesan dari aplikasi klien	Mengirim notification, chat, dan pesan lain dari perangkat ke server melalui saluran koneksi FCM

6.1.2 Cara Kerja Firebase Cloud Messaging

Implementasi FCM mencakup dua komponen utama untuk mengirim dan menerima pesan:

- Lingkungan tepercaya seperti Cloud Functions untuk Firebase atau server aplikasi yang akan digunakan untuk membuat, menargetkan, dan mengirim pesan.
- Aplikasi klien iOS, Android, atau Web (JavaScript) yang menerima pesan.

Anda dapat mengirim pesan melalui Admin SDK atau API HTTP dan XMPP. Anda juga dapat menggunakan Notifications Composer untuk menguji atau mengirim pesan pemasaran atau interaksi dengan penargetan dan analisis bawaan yang andal. Alur pengiriman pesan melalui FCM ditunjukkan oleh Gambar 6.1

Gambar 6.1 Alur pengiriman pesan menggunakan FCM (sumber gambar : https://firebase.google.com/docs/cloud-messaging)

6.1.3 Alur Implementasi Firebase Cloud Messaging

1. Menyiapkan FCM SDK	Menyiapkan Firebase dan FCM pada aplikasi sesuai petunjuk penyiapan untuk platform Anda.
2. Mengembangkan aplikasi klien	Menambahkan penanganan pesan, logika langganan topik, atau fitur opsional lain ke aplikasi klien Anda. Selama pengembangan, Anda dapat mengirim pesan pengujian dengan mudah dari Notifications Composer
3. Mengembangkan server aplikasi	Tentukan apakah Anda ingin menggunakan Admin SDK atau salah satu protokol server untuk membuat logika pengiriman, yaitu logika untuk mengautentikasi, membuat permintaan pengiriman, menangani respons, dan sebagainya. Kemudian, buat logika di lingkungan yang tepercaya.

6.2 Jenis Pesan pada Firebase Cloud Messaging

Dengan FCM, Anda dapat mengirim 2 jenis pesan ke klien:

- Notification message, terkadang dianggap sebagai "pesan tampilan".
- Data message, yang ditangani oleh aplikasi klien.

Pesan notification memiliki serangkaian kunci bawaan yang terlihat oleh pengguna. Sebaliknya, pesan data hanya memuat *key-value pair* khusus. Berikut contoh dan penjelasan masing-masing jenis pesan di FCM

6.2.1 Notification Message

Untuk pengujian atau pemasaran dan penarik interaksi pengguna, Anda dapat mengirim pesan *notification* menggunakan Notifications Composer. Untuk mengirim pesan notification secara terprogram menggunakan Admin SDK atau protokol FCM, setel kunci notification dengan rangkaian opsi key-value bawaan yang diperlukan untuk bagian pesan notification yang terlihat oleh pengguna. Misalnya, berikut ini pesan notification berformat JSON dalam aplikasi IM. Pengguna akan menemukan pesan dengan judul "Pesan untuk anda" dan teks "Selamat pagi dunia" di perangkat:

```
{
   "to" : "/topics/pjr9",
   "notification" : {
          "body" : "Selamat pagi dunia",
          "title" : "Pesan untuk anda"
   }
}
```

6.2.2 Data Message

Setel kunci data dengan *key-value pair* khusus Anda untuk mengirim payload data ke aplikasi klien. Pesan data dapat memiliki payload maksimal 4 KB.Misalnya, berikut adalah pesan berformat JSON dalam aplikasi instant messaging yang mengirimkan data (payload) yang diharapkan dapat diproses oleh aplikasi klien :

```
{
   "to" : "/topics/pjr9",
   "data" : {
          "judul" : "Produk Baru",
          "id" : "P6712",
          "nama_brg" : "Obat Nyamuk Rasa Madu"
   }
}
```

6.2.3 Pesan Gabungan Notification dan Data

Baik secara terprogram atau melalui Firebase console, Anda dapat mengirim pesan notification yang berisi payload opsional key-value pair khusus. Di penulis Notifications, gunakan kolom Data khusus pada Opsi lanjutan. Perilaku aplikasi saat menerima pesan yang memuat payload notification dan data bergantung pada apakah aplikasi itu berjalan di latar belakang atau latar depan - intinya, apakah aplikasi aktif atau tidak pada saat menerima pesan.

- Saat berjalan di latar belakang, aplikasi menerima payload notification di baki notification, dan hanya menangani payload data saat pengguna klik pada notification yang masuk.
- Saat berjalan di latar depan, aplikasi Anda menerima objek pesan dengan kedua payload yang tersedia.

Berikut adalah pesan berformat JSON yang memuat kunci notification dan juga

kunci data:

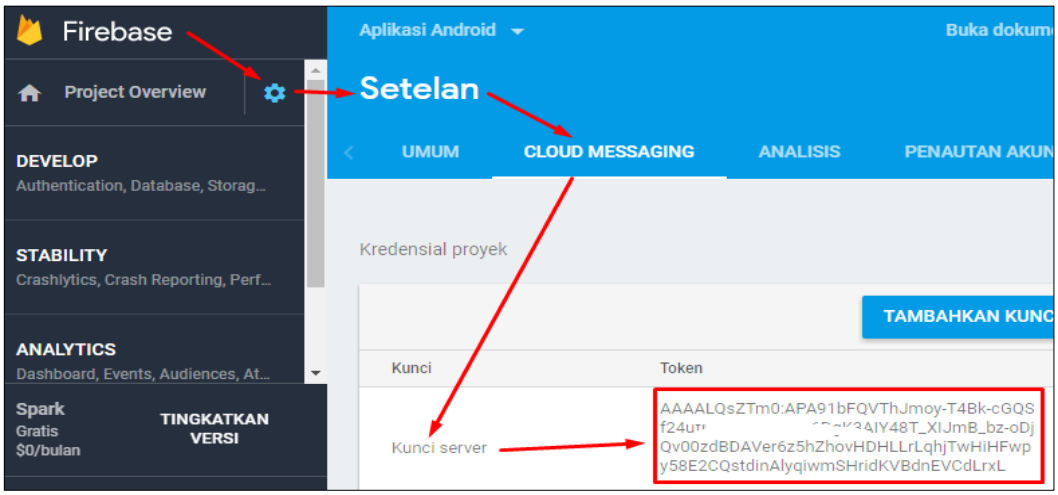
```
{
   "to" : "/topics/pjr9",
   "notification" : {
          "body" : "Selamat pagi dunia",
          "title" : "Pesan untuk anda"
   },
   "data" : {
          "id" : "P6712",
          "nama_brg" : "Obat Nyamuk Rasa Madu"
   }
}
```

6.3 Praktik Pembuatan Aplikasi

1. Daftarkan **nama package** dari project aplikasi anda ke Google Firebase
2. Download file **google-services.json** ke dalam project anda
3. Pada Firebase Console, pilih menu **Setting/Setelan → CLOUD MESSAGING.** Dari CLOUD MESSAGING salin/copy token Kunci Server seperti ditunjukkan oleh Gamabr 6.2. Token Kunci Server ini nantinya dapat digunakan untuk mengautentikasi/mengenali aplikasi server yang Anda gunakan untuk mengirim pesan ke aplikasi client. Setiap pesan yang Anda kirim dari aplikasi server Anda ke aplikasi klien harus melalui server Firebase yang berada pada alamat :

   https://fcm.googleapis.com/fcm/send

   Pesan yang dikirim melalui aplikasi server Anda harus menggunakan protokol HTTP metode POST dengan header pesan/message berisi token Kunci Server yang terdaftar di dalam proyek Firebase yang Anda buat



Gambar 6.2 Salin server key FCM untuk digunakan pada server Anda(sumber gambar : https://console.firebase.google.com)

6.3.1 File build.gradle (Project)

```
buildscript {
    dependencies {
        classpath 'com.google.gms:google-services:4.4.0'
    }
}// Top-level build file where you can add configuration options common to
all sub-projects/modules.
plugins {
    id 'com.android.application' version '8.1.2' apply false
    id 'org.jetbrains.kotlin.android' version '1.9.0' apply false
}
```

6.3.2 File build.gradle (Module)

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'com.google.gms.google-services'
}

android {
    compileSdk 34

    defaultConfig {
        applicationId "polinema.mi.appx0f"
        minSdk 21
        targetSdk 34
        versionCode 1
```

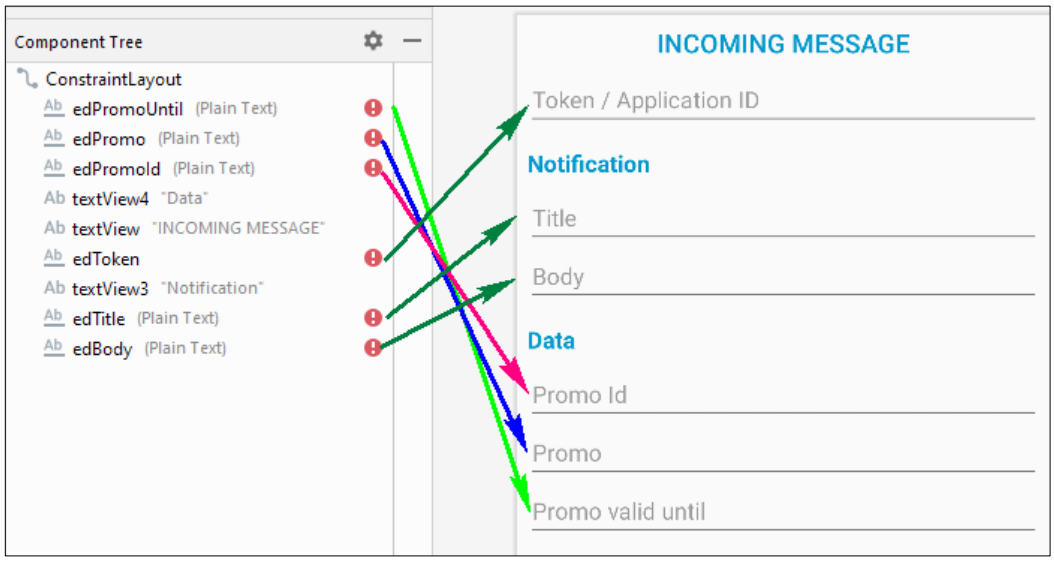
```
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile(
                    'proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    buildFeatures{
        viewBinding true
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    namespace 'polinema.mi.appx0f'
}

dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'androidx.core:core-ktx:1.12.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    implementation platform(
                    'com.google.firebase:firebase-bom:28.4.1')
    implementation 'com.google.firebase:firebase-messaging-ktx'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test:runner:1.5.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

6.3.3 Membuat antarmuka aplikasi

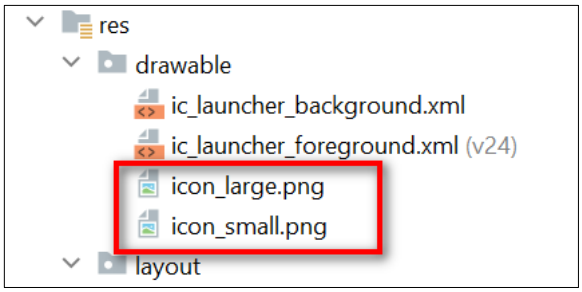
Antarmuka aplikasi **(activity_main.xml)** berguna untuk menampilkan pesan berupa notifikasi atau data. Rancangan antarmuka pengguna dapat dilihat pada Gambar 6.3



Gambar 6.3 Rancangan antarmuka pengguna (activity_main.xml)

6.3.4 Menambahkan obyek drawable

Tambahkan dua buah file drawable berupa icon untuk ditampilkan pada message yang masuk seperti ditunjukkan oleh Gambar 4. ***Icon_large.png*** berukuran 192x192 pixel sementara ***icon_small.png*** berukuran 72x72 pixel



Gambar 6.4 menambahkan obyek drawable

6.3.5 File MainActivity.kt

Class ini digunakan untuk menampilkan pesan/message yang masuk, baik pesan berupa notifikasi maupun data, dan dapat digunakan untuk memproses pesan lebih lanjut.

```
package polinema.mi.appx0f

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.widget.Toast
import com.google.firebase.ktx.Firebase
import com.google.firebase.messaging.FirebaseMessaging
import com.google.firebase.messaging.ktx.messaging
import polinema.mi.appx0f.databinding.ActivityMainBinding
```

```
import java.lang.Exception

class MainActivity : AppCompatActivity(){

    lateinit var b : ActivityMainBinding
    var bundle : Bundle? = null
    var topik = "appx0f"
    var type = 0

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        b = ActivityMainBinding.inflate(layoutInflater)
        setContentView(b.root)
        Firebase.messaging.subscribeToTopic(topik)
        .addOnCompleteListener {
            var msg = "Subcribe to $topik"
            if(!it.isSuccessful) msg = "Can't subscribe to topic"
            Toast.makeText(this, msg, Toast.LENGTH_LONG).show()
        }
    }

    override fun onResume() {
        super.onResume()

        FirebaseMessaging.getInstance().token.addOnCompleteListener {
            if(!it.isSuccessful) return@addOnCompleteListener
            b.edToken.setText(it.result!!.toString())
        }

        try {
            bundle = getIntent().getExtras()!!
        }catch (e : Exception){
            Log.e("BUNDLE","bundle is null")
        }

        if(bundle != null){
            type = bundle!!.getInt("type")
            when(type){
                0 -> {
                    b.edPromoId.setText(bundle!!.getString("promoId"))
                    b.edPromo.setText(bundle!!.getString("promo"))
                    b.edPromoUntil.setText(bundle!!.getString("promoUntil"))
                }
                1 -> {
                    b.edTitle.setText(bundle!!.getString("title"))
                    b.edBody.setText(bundle!!.getString("body"))
                }
            }
        }
    }
}
```

6.3.6 File Appx0fFirebaseMessagingService.kt

Class ini berfungsi sebagai service yang menerima pesan/message dan menampilkan notifikasi ke layar smartphone pengguna.

```
package polinema.mi.appx0f

import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.graphics.BitmapFactory
import android.media.AudioAttributes
import android.media.RingtoneManager
import android.os.Build
import androidx.core.app.NotificationCompat
import com.google.firebase.messaging.FirebaseMessagingService
import com.google.firebase.messaging.RemoteMessage

class Appx0fFirebaseMessagingService : FirebaseMessagingService() {
    var promoId = ""
    var promo = ""
    var promoUntil = ""
    var body = ""
    var title = ""
    val RC_INTENT = 100
    val CHANNEL_ID = "appx0f"

    override fun onNewToken(p0: String) {
        super.onNewToken(p0)
    }

    override fun onMessageReceived(p0: RemoteMessage) {
        super.onMessageReceived(p0)

        val intent = Intent(this, MainActivity::class.java)
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP)

        //if the incoming message is a data, get its content
        if(p0.getData().size > 0){
            promoId = p0.data.getValue("promoId")
            promo = p0.data.getValue("promo")
            promoUntil = p0.data.getValue("promoUntil")

            intent.putExtra("promoId",promoId)
            intent.putExtra("promo", promo)
            intent.putExtra("promoUntil",promoUntil)
            intent.putExtra("type",0) //0 = data

            sendNotif("Today Promo!!!", "$promo $promoUntil", intent)
        }
```

```
        //if the incoming message is a notification, get its title and body
        if(p0.notification != null){
            body = p0.notification!!.body!!
            title = p0.notification!!.title!!

            intent.putExtra("title",title)
            intent.putExtra("body", body)
            intent.putExtra("type",1) //1 = notification

            sendNotif(title, body, intent)
        }
    }

    //this function will be called when there is a data message
    fun sendNotif(title : String, body : String, intent : Intent){
        //this pendingIntent will be called when there is an incoming message
        //and invoked the predefined intent
        val pendingIntent =
            PendingIntent.getActivity(this, RC_INTENT, intent,
            PendingIntent.FLAG_ONE_SHOT or
            PendingIntent.FLAG_IMMUTABLE
        )

        //choose a ringtone
        val ringtoneUri = RingtoneManager.getDefaultUri(
            RingtoneManager.TYPE_RINGTONE)
        val audioAttributes = AudioAttributes.Builder().
            setUsage(AudioAttributes.USAGE_NOTIFICATION_RINGTONE).
            setContentType(
                    AudioAttributes.CONTENT_TYPE_SONIFICATION).build()

        //create a notification manager
        val notificationManager =
            getSystemService(Context.NOTIFICATION_SERVICE) as
            NotificationManager

        //When we target Android 8.0 (API level 26),
        //we must implement one or more notification channels
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val mChannel = NotificationChannel(CHANNEL_ID, "appx0f",
                NotificationManager.IMPORTANCE_HIGH )
            mChannel.description = "App x0f"
            mChannel.setSound(ringtoneUri,audioAttributes)
            notificationManager.createNotificationChannel(mChannel)
        }

        //build a notification
        val notificationBuilder =
            NotificationCompat.Builder(baseContext,CHANNEL_ID)
            .setSmallIcon(R.drawable.icon_small)
            .setLargeIcon(BitmapFactory.decodeResource(
```

```
                    resources,R.drawable.icon_large))
            .setContentTitle(title)
            .setContentText(body)
            .setSound(ringtoneUri)
            .setContentIntent(pendingIntent)
            .setAutoCancel(true).build()
        notificationManager.notify(RC_INTENT,notificationBuilder)
    }
}
```

6.3.7 File AndroidManifest.xml

Pada **AndroidManifest.xml** harus didefinisikan class yang berperan sebagai Service. Service merupakan aplikasi yang berjalan di background dan tidak ditampilkan secara visual layaknya Activity. Service akan dijalankan ketika terjadi event tertentu, misal ketika ada pesan masuk. Aplikasi yang mengimplementasikan FCM membutuhkan sebuah service, yaitu

- **Firebase Messaging Service,** untuk menerima pesan/message yang masuk dan melakukan operasi lebih lanjut terhadap pesan yang masuk serta membuat token ID aplikasi klien. Token ID akan selalu berbeda pada setiap handphone yang menjalankan aplikasi klien FCM

Berdasarkan penjelasan diatas, berikut implementasi dari **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission
          android:name="android.permission.POST_NOTIFICATIONS" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <meta-data
            android:name=
                "com.google.firebase.messaging.default_notification_icon"
            android:resource="@android:drawable/ic_dialog_email" />

        <meta-data
            android:name=
                "com.google.firebase.messaging.default_notification_color"
            android:resource="@color/colorPrimary" />

        <meta-data
            android:name=
```

```
            "com.google.firebase.messaging.default_notification_channel_id"
          android:value="appx0f" />

       <service android:name=".Appx0fFirebaseMessagingService"
          android:exported="false">
          <intent-filter>
             <action android:name=
                   "com.google.firebase.MESSAGING_EVENT" />
          </intent-filter>
       </service>

       <activity android:name=".MainActivity"
          android:exported="true">
          <intent-filter>
             <action android:name="android.intent.action.MAIN" />

             <category android:name=
                                  "android.intent.category.LAUNCHER" />
          </intent-filter>
       </activity>

    </application>
</manifest>
```

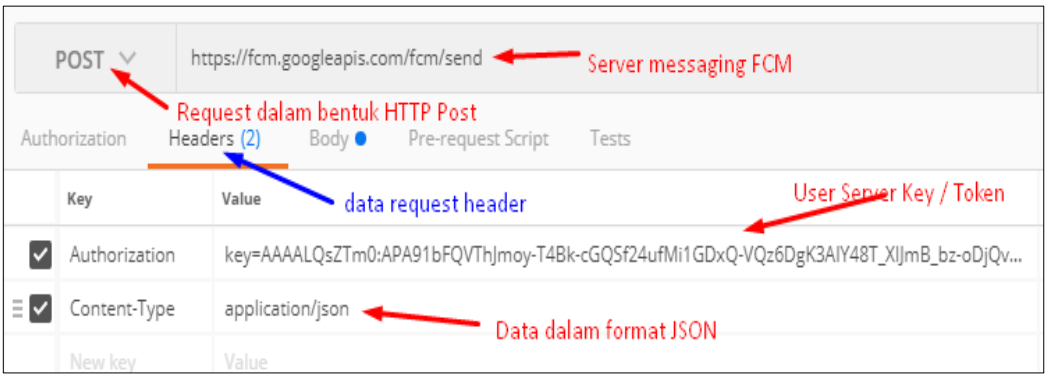
6.4 Uji Coba Aplikasi

Download dan install aplikasi POSTMAN untuk melakukan uji coba mengirim pesan dari Server (Gambar 6.5). Anggaplah POSTMAN sebagai aplikasi server Anda yang dapat bertugas mengirim pesan menggunakan protokol HTTP

Gambar 6.5 Aplikasi POSTMAN (sumber gambar : https://www.getpostman.com)

6.4.1 Konfigurasi Header Pengiriman Pesan ke Server FCM

Agar message/pesan yang Anda kirim ke klien melalui Server FCM dapat diproses dengan semestinya, maka Anda perlu menyertakan Server Key proyek aplikasi Anda ke dalam data header dari pesan yang akan dikirim. Berikut konfigurasi yang perlu dituliskan ke dalam POSTMAN (Gambar 6.6)



Gambar 6.6 Konfigurasi message header dari pesan/message FCM

6.4.2 Mengirim Pesan Notification

Coba lakukan uji coba mengirim pesan notification ke aplikasi klien dan lihat perbedaan bagaimana pesan diterima ketika : 1) aplikasi klien FCM berjalan di foreground, dan, 2) aplikasi klien berjalan di background. Untuk mengirim pesan notification, pada bagian Body pada POSTMAN, ketikkan pesan dalam format JSON seperti pada Gambar 6.7

Gambar 6.7 Body message berupa notification dari pesan yang akan dikirim menggunakan FCM

6.4.3 Mengirim Pesan Data

Coba lakukan uji coba mengirim pesan data ke aplikasi klien dan lihat perbedaan bagaimana pesan diterima ketika : 1) aplikasi klien FCM berjalan di foreground, dan, 2) aplikasi klien berjalan di background. Untuk mengirim pesan data, pada bagian Body pada POSTMAN, ketikkan pesan dalam format JSON sebagai berikut :

Gambar 6.8 Body message berupa data dari pesan yang akan dikirim menggunakan FCM

6.4.4 Mengirim Pesan Secara Private ke Perangkat Tertentu

Selain digunakan untuk brodcast pesan, FCM juga dapat digunakan untuk mengirim pesan ke salah satu atau beberapa klien. Caranya dengan mengirim pesan ke klien dengan Token ID tertentu, contoh Token ID seperti Gambar 6.9. Dengan cara ini hanya klien yang memiliki Token ID tersebut yang bisa menerima pesan

Gambar 6.9 Body message berupa data dari pesan yang akan dikirim secara private menggunakan FCM

Referensi :

https://firebase.google.com/docs/cloud-messaging/android/client