



JavocSoft-ToolBox: Modulo de Notificaciones

Indice

[ACTIVAR GCM PARA NUESTRA APLICACIÓN ANDROID](#)

[Dar de alta el proyecto en Google Cloud Console](#)

[Activar GCM](#)

[Registro de la aplicación Android](#)

[Dar de alta la parte servidora](#)

[En resumen](#)

[Más información](#)

[Uso del módulo de notificaciones de la librería JavocSoft-ToolBox](#)

[Integración](#)

[1. Añadir los permisos y artefactos necesarios al AndroidManifest.xml](#)

[2. Crear las variables del módulo de notificaciones de la librería y el receiver de notificaciones](#)

[3. Inicializar el módulo de notificaciones y el receiver](#)

[4. Registrar el dispositivo con GCM](#)

[5. Recoger las notificaciones PUSH cuando lleguen](#)

[6. Desregistrar el dispositivo si se sale de la aplicación](#)

[Eventos, interacción](#)

[Envío de Notificaciones](#)

[Tipos de mensajes](#)

[Send-to-sync messages \(Collapsible\)](#)

[Messages with payload \(Non-Collapsible\)](#)

[Parametros del mensaje](#)

[Time to Live \(time_to_live\)](#)

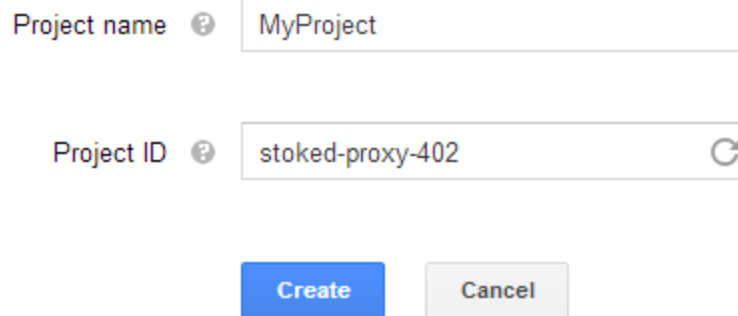
[Delay While IDLE \(delay_while_idle\)](#)

ACTIVAR GCM PARA NUESTRA APLICACIÓN ANDROID

Dar de alta el proyecto en Google Cloud Console

Antes de nada, hemos de dar de alta el proyecto en Google Cloud Console. Para ello ir a la dirección <https://cloud.google.com/console>. dadle al botón “Create project” y rellenad el pop-up que se os muestra:

New Project

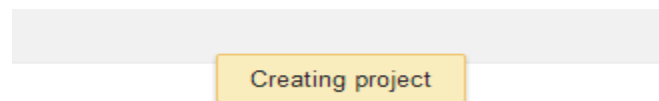


Project name ? MyProject

Project ID ? stoked-proxy-402

Create Cancel

Dadle al botón “Create”, es posible que tarde un poco, en este caso vereis un pop-up pequeño indicándo el proceso:



Bien, ya tenemos nuestro proyecto dado de alta en el Cloud de Google. Esto es necesario dado que vamos a usar una API de Google y Google requiere esto para poder configurar y usar sus APIs.

Ahora mismo os encontrareis con una pantalla tal que



Lo importante de esta pantalla es el **Project Number** que se indica en la esquina superior derecha de la imagen. Este número será el **SenderId** usado por nuestra aplicación Android para registrarse en el sistema de notificaciones PUSH de Google llamado **GCM**.

Activar GCM

A continuación, hemos de activar dicho servicio (GCM). Nos vamos a APIs y buscamos el nombre "Google Cloud Messaging for Android", esta es la única que nos interesa que este "ON".



< MyProject

Overview

APIs & auth

APIs

Registered apps

Consent screen

Notification endpoints

Permissions

Settings

Support

NAME

[Google Cloud Messaging for Android](#)

[Ad Exchange Buyer API](#)

[Ad Exchange Seller API](#)

[Admin SDK](#)

[AdSense Host API](#)

[AdSense Management API](#)

[Analytics API](#)

Registro de la aplicación Android

Una vez hecho esto, registramos la aplicación Android que usará el GCM activado para nuestro proyecto. Nos vamos a "Registered apps" y damos al botón rojo "Register App".

Google Cloud Console

< MyProject

REGISTER APP

Overview

APIs & auth

APIs

Registered apps

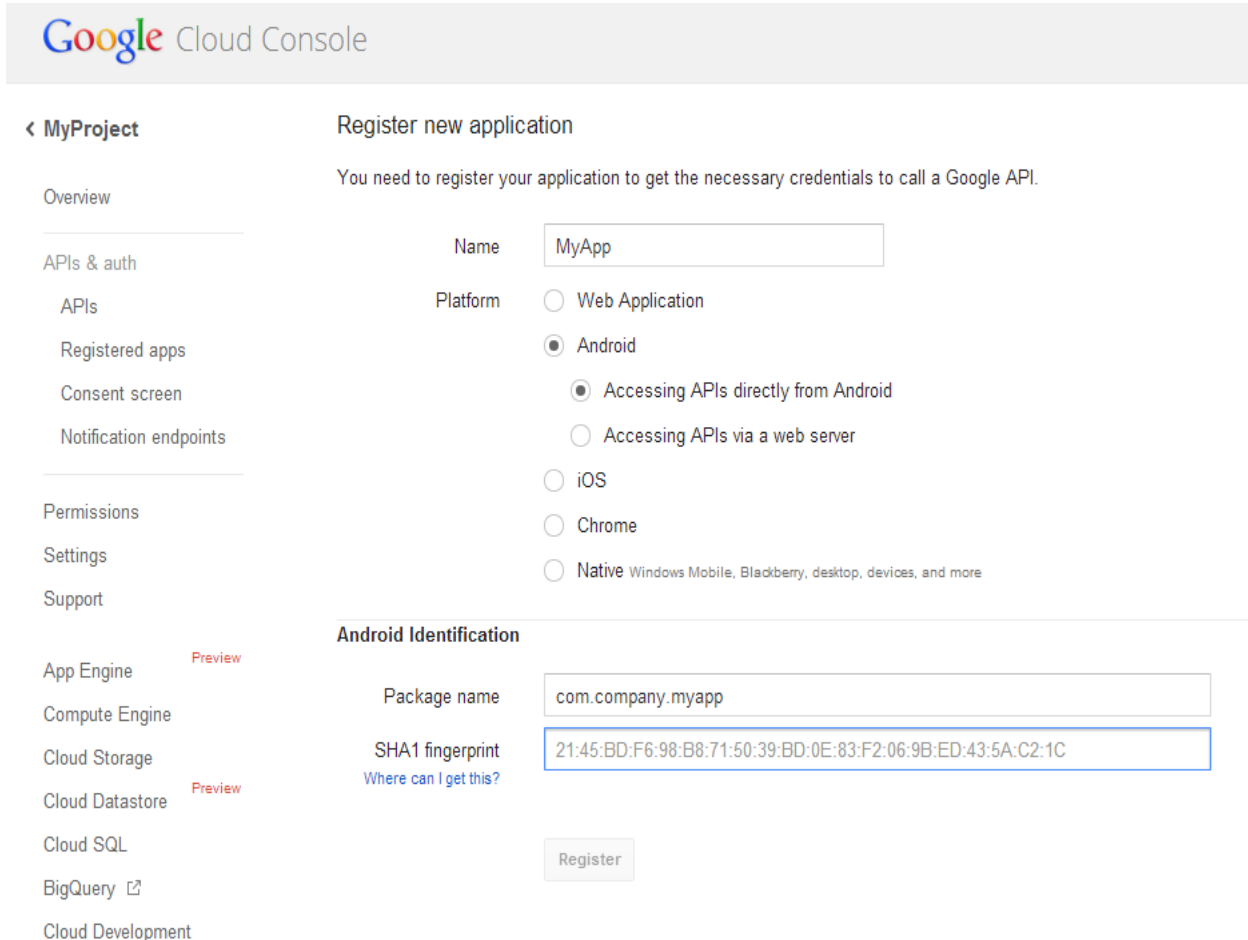
Consent screen

Notification endpoints

NAME

[Service Account-project](#)

Al hacerlo se nos mostrará otra ventana, la rellenamos:



De esta ventana indicamos el nombre, el package y que la app es para Android y el resto lo dejamos como en la imagen. El campo SHA1 lo obtenemos ejecutando

To find your SHA-1 fingerprint:

1. In a terminal, run the following [Keytool](#) command to generate the signing certificate's SHA-1 fingerprint:

```
keytool -exportcert -alias androiddebugkey -keystore {path-to-debug-or-production-keystore} -list -v
```

Note: For Eclipse, the debug keystore is typically located at `~/android/debug.keystore`.

2. Keytool prompts you to enter a password for the keystore. The default password for the debug keystore is `android`. Keytool then prints the fingerprint to the shell. For example:

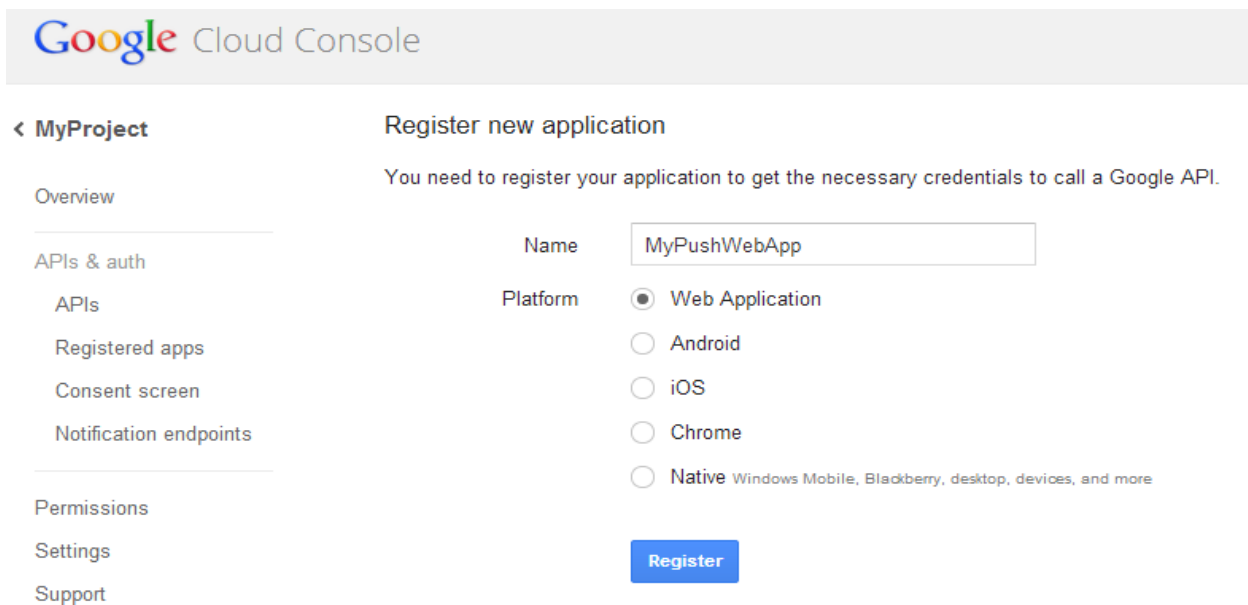
```
Certificate fingerprint: SHA1: DA:39:A3:EE:5E:6B:4B:0D:32:55:BF:EF:95:60:18:90:AF:D8:07:09
```

Tal como indica en <https://developers.google.com/console/help/new/#installedapplications>.

Con esto ya tenemos lo necesario para que la aplicación Android se pueda registrar y usar el servicio GCM.

Dar de alta la parte servidora

Bien, ahora nos toca la parte servidora, es decir, desde donde vamos a enviar las notificaciones PUSH. Nos vamos de nuevo a “Registered Apps” y creamos una nueva dándole de nuevo al botón rojo “Register App”. En este caso indicamos el tipo “Web Application”, ponemos un nombre y le damos al botón azul “Registrar”.



Una vez termine de crearse la aplicación nos vamos dentro de ella. De las cuatro secciones de que dispone la que nos interesa es la que se llama “**Browser key**”, en concreto el parámetro “API KEY”. Este será el valor que usaremos en el servidor para poder enviar mensajes PUSH a

▼ **Browser Key**

Access data that comes from a browser, and that is not associated with an account

API KEY

ALLOWED REFERRERS

Any referrer is allowed. Restrict access

ACTIVATED ON

Nov 17, 2013 9:11 AM

ACTIVATED BY

javocsoft@gmail.com - you

la aplicación.

En resumen

- Dar de alta el proyecto para obtener el SenderID necesario por la aplicación Android para registrarse y usar el servicio GCM.
- Dar de alta la aplicación Android.
- dar de alta la aplicación web/programa que va a enviar las pushes.

Más información

<http://developer.android.com/google/gcm/gs.html>

Uso del módulo de notificaciones de la librería JavocSoft-ToolBox

Integración

Los pasos son los siguientes:



1. Añadir los permisos y artefactos necesarios al AndroidManifest.xml

Permisos:

```
<permission android:name="es.javocsoft.test.gcm.permission.C2D_MESSAGE"
            android:protectionLevel="signature" />
<uses-permission android:name="es.javocsoft.test.gcm.permission.C2D_MESSAGE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

Resto de artefactos necesarios, dentro de la etiqueta "application":

```
<receiver
    android:name="es.javocsoft.android.lib.toolbox.gcm.core.CustomGCMBroadcastReceiver"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <category android:name="es.javocsoft.test.gcm" />
    </intent-filter>
</receiver>
<service android:name="es.javocsoft.android.lib.toolbox.gcm.core.GCMIntentService" />
```

2. Crear las variables del módulo de notificaciones de la librería y el receiver de notificaciones

```
private NotificationModule notificationModule= null;
private CustomNotificationReceiver notReceiver = null;
```

3. Inicializar el módulo de notificaciones y el receiver

En el método onCreate():

```
notificationModule = NotificationModule.getInstance(this, EnvironmentType.SANDBOX,
    appGCMSenderId,
    new AfterRegistration(), new OnAck(), null,
    new OnNotReceived(),
    true,null);

if(notReceiver==null){
    notReceiver = new CustomNotificationReceiver();
    registerReceiver(notReceiver, new
        IntentFilter(NotificationModule.NEW_NOTIFICATION_ACTION));
}
```




```
}
```

4.Registrar el dispositivo con GCM

En el método onResume():

```
notificationModule.gcmRegisterDevice(this, "Testing GCM", MainActivity.class);
```

5.Recoger las notificaciones PUSH cuando llegan

En el método onNewIntent():

```
notificationModule.gcmCheckForNotificationReceival(this, intent);
```

6.Desregistrar el dispositivo si se sale de la aplicación

En el método onDestroy() ó en el botón que tengamos para salir de la aplicación:

```
unregisterReceiver(notReceiver);  
notificationModule.gcmUnregisterDevice(this, null);
```

Tras aplicar todo esto a la clase Main de nuestro proyecto queda tal que:

```
1 package es.javocsoft.test.gcm;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.content.IntentFilter;
6 import android.os.Bundle;
7 import es.javocsoft.android.lib.toolbox.gcm.EnvironmentType;
8 import es.javocsoft.android.lib.toolbox.gcm.NotificationModule;
9 import es.javocsoft.android.lib.toolbox.gcm.core.CustomNotificationReceiver;
10
11 public class MainActivity extends Activity {
12
13     private final String appGCMSenderId = "<appSenderId>"; //<-- Aquí el Project Number
14
15     private final String NOTIFICATION_TITLE = "Testing GCM";
16     private NotificationModule notificationModule= null;
17     private CustomNotificationReceiver notReceiver = null;
18
19     @Override
20     protected void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.activity_main);
23
24         notificationModule = NotificationModule.getInstance(this, EnvironmentType.SANDBOX, appGCMSenderId,
25             new AfterRegistration(), new OnAck(), null,
26             new OnNotReceived(),
27             true,null);
28
29         if(notReceiver==null){
30             notReceiver = new CustomNotificationReceiver();
31             registerReceiver(notReceiver, new IntentFilter(NotificationModule.NEW_NOTIFICATION_ACTION));
32         }
33     }
34
35     @Override
36     protected void onResume() {
37         super.onResume();
38         notificationModule.gcmRegisterDevice(this, NOTIFICATION_TITLE, MainActivity.class);
39     }
40
41     @Override
42     protected void onNewIntent(Intent intent) {
43         super.onNewIntent(intent);
44         notificationModule.gcmCheckForNotificationReceival(this, intent);
45     }
46
47     @Override
48     protected void onDestroy() {
49         super.onDestroy();
50
51         unregisterReceiver(notReceiver);
52         notificationModule.gcmUnregisterDevice(this, null);
53     }
54 }
55
```

Sencillo, no?

Eventos, interacción

La librería dispone de unas clases que permiten que puedas ejecutar el código que necesites cuando suceden eventos relacionados con las notificaciones de GCM. En concreto, cuando:



- Se produce el registro, mediante:
 - Clase **GCMIntentService.OnRegistrationRunnableTask**
- Cuando se recibe una notificación, mediante:
 - Clase **CustomNotificationReceiver.OnNewNotificationRunnableTask**
- Cuando el usuario abre la notificación, mediante:
 - Clase **CustomNotificationReceiver.OnAckRunnableTask**

Las tres clases proveen de tres métodos a sobrescribir:

```
@Override
protected void pre_task() {}

@Override
protected void task() {}

@Override
protected void post_task() {}
```

Además, proveen también de lo que puedas necesitar:

- La clase GCMIntentService.OnRegistrationRunnableTask te deja consultar el token de registro de GCM a través del método getGCMRegistrationToken() para que puedas usarlo y dar de alta dicho id en tu sistema de backend. *Devolverá null si no se pudo realizar el registro.*
- Las clases OnNewNotificationRunnableTask y OnAckRunnableTask de *CustomNotificationReceiver* te permiten acceder a los datos de la notificación a través del método getExtras(). Por lo que puedes hacer lo que requieras.

Un ejemplo de uso sería el enviar este id junto con algún dato más a nuestro sistema de backend para así poder registrar en nuestro sistema los dispositivos y poder más adelante enviarles pushes.

```
1 package es.javocsoft.test.gcm;
2
3 import android.util.Log;
4 import es.javocsoft.android.lib.toolbox.gcm.core.GCMIntentService;
5
6 public class AfterRegistration extends GCMIntentService.OnRegistrationRunnableTask{
7
8     @Override
9     protected void pre_task() {
10
11     }
12
13     @Override
14     protected void task() {
15         Log.i("GCMTest", getGCMRegistrationToken());
16     }
17
18     @Override
19     protected void post_task() {
20
21     }
22 }
23
```

Envío de Notificaciones

La parte servidora, solo la parte del envío sería algo tan sencillo como:

```
package es.javocsoft.gcm.server.test;

import java.util.ArrayList;
import java.util.Random;

import com.google.android.gcm.server.Message;
import com.google.android.gcm.server.Message.Builder;
import com.google.android.gcm.server.MulticastResult;
import com.google.android.gcm.server.Sender;

public class SendPush {

    /**
     * @param args
     */
    public static void main(String[] args) {

        //API access (Google API Console - Web Browser application -> Browser Key)
        final String apiKey = "apikey";
        //The android device registration GCM registration id token
        final String devRegId = "gcmRegtoken";
```

```
try{
    ArrayList<String> devicesList = new ArrayList<String>();
    devicesList.add(devRegId);

    Sender sender = new Sender(apiKey);
    Builder msgBuilder = new Message.Builder();
    //See http://developer.android.com/google/gcm/server.html for options.
    //msgBuilder.delayWhileIdle(true);
    //msgBuilder.collapseKey("collapseKey");
    //msgBuilder.timeToLive(3);
    msgBuilder.addData("message", "Hola que tal? " + new Random().nextInt());
    Message message = msgBuilder.build();

    //Just for one device.
    /*Result result = sender.send(message, devRegId, 5);
    if (result.getMessageId() != null) {
        String canonicalRegId = result.getCanonicalRegistrationId();
        if (canonicalRegId != null) {
            // same device has more than on registration ID: update database
        }
    } else {
        String error = result.getErrorCodeName();
        if (error.equals(Constants.ERROR_NOT_REGISTERED)) {
            // application has been removed from device - unregister database
        }
    }
    */

    //For multiple devices
    MulticastResult result = sender.send(message, devicesList, 1);
    System.out.println(result.toString());
    if (result.getResults() != null) {
        int canonicalRegId = result.getCanonicalIds();
        if (canonicalRegId != 0) {
        }
    } else {
        int error = result.getFailure();
        System.out.println(error);
    }

} catch (Exception e) {
    e.printStackTrace();
}
}
```

Tipos de mensajes

Existen dos tipos de mensajes:



Send-to-sync messages (Collapsible)

Son mensajes tipo ping para notificar a la aplicación que ha pasado algo. Son mensajes en los que no nos hace falta recibir cada notificación sino la última.

Por ejemplo si la app recibe correos, no nos interesa que en la app se muestre 5 notificaciones de nuevo mail si se hace la verificación cada minuto, por ejemplo, sino una sola que indique que hay nuevo correo.

Para ello indicamos en el envío `collapse_key` a un valor (texto que agrupe al conjunto de mensajes de algo). GCM permite hasta un total de 4 `collapse_key`. Si nos pasamos del tope de claves GCM no nos asegurará el valor que tendrán al llegar al cliente.

Este tipo de mensajes tienen un alto rendimiento.

Messages with payload (Non-Collapsible)

Son mensajes con un contenido que puede ser todo lo complejo que se deseé. Es un JSON insertado en el campo "data" del JSON de notificación GCM.

En este caso cada notificación nos importa así que no usamos `collapse_key` ya que nos interesa recibir todas las notificaciones. Es importante decir que:

- GCM no asegura el orden de envío.
- GCM almacenará hasta un máximo de 100 mensajes de este tipo. Mas allá de este límite GCM enviará un mensaje de tipo Collapsible en donde:
 - `message_type` Tendrá el texto "deleted_messages".
 - `total_deleted` Indicará el número de eliminados.

En este caso la aplicación debería sincronizarse con el servidor para bajarse todos los pendientes y así evitar este error.

Parametros del mensaje



Time to Live (**time_to_live**)

Este parámetro permite indicar, en segundos (máximo 2,419,200), el tiempo de vida de un mensaje. Por defecto son 4 semanas si no se especifica. Durante este tiempo GCM tratará de enviar el mensaje.

Indicamos un valor de 0 haremos que si GCM no puede enviar el mensaje lo descarte, es como un “ahora o nunca”.

Delay While IDLE (**delay_while_idle**)

Estableciendo este valor a **true** haremos que GCM espere a que el dispositivo este activo (con cobertura, encendido, etc) antes de tratar de enviar la notificación.