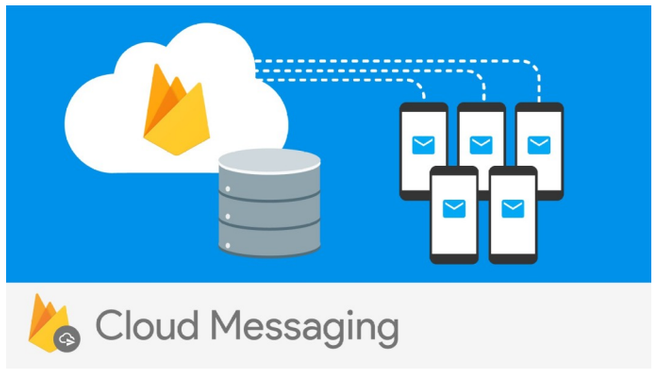
Servicios de las plataformas móviles

Sesión 1: Servicios de autenticación y notificaciones

Puntos a tratar

- *Autenticación de usuarios con Google Sign In*
- *Notificaciones push*



Firebase cloud messaging

Visión general

- ¿Qué es FCM?
  - Es un servicio de mensajes
  - Es multiplataforma
  - Es gratuito.
  - El FCM se encarga de toda la gestión.
  - Tienen una capacidad máxima de 4KB.

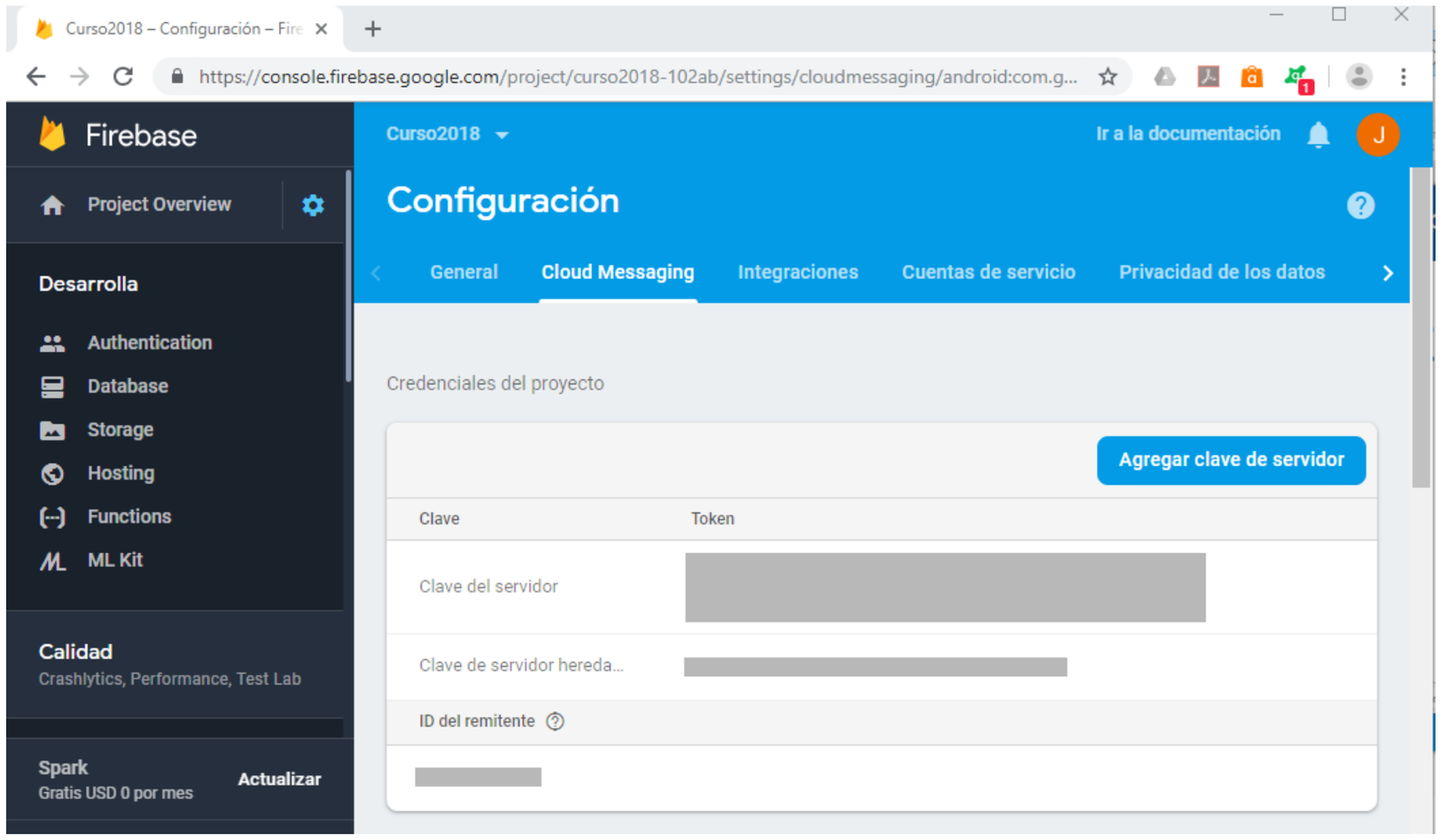
Funciones clave

- Permite envío de mensajes a nuestra aplicación de tres formas distintas:
  - A un dispositivo individual.
  - A un grupo de dispositivos.
  - A los dispositivos suscritos a un tema.
- Permite el envío de dos tipos de mensajes:
  - **Mensajes de notificación**: El SDK de FCM maneja automáticamente estos mensajes. Contienen un conjunto predefinido de claves visibles para el usuario.
  - **Mensajes de datos**: Los gestiona la aplicación cliente. Contienen pares clave-valor personalizados definidos por el usuario.
- Permite el envío de mensajes desde la aplicación cliente al servidor.

Arquitectura

Conceptos clave

- Componentes.
  - Aplicación cliente.
  - Aplicación de servidor.
  - Servidores de conexión de FCM.
- Credenciales.
  - ID del remitente.
  - Token de registro.
  - Clave del servidor.



Pasos a realizar para implementar FCM

1. Desarrollar nuestra **aplicación cliente**.
   - Configurar Firebase y el FCM SDK.
   - Agregar administración de mensajes, lógica de suscripción a temas y otras funciones opcionales a nuestra app cliente.
2. Implementar una aplicación de servidor para interaccionar con el servidor de conexión FCM elegido.

Creación de una aplicación cliente

- Prerequisitos de la plataforma FCM para Android.
- Habilitación de la API de FCM.
- Implementación de la aplicación cliente.
  - Configuración del proyecto.
  - Configuración de los servicios (opcionales) a utilizar por nuestra aplicación:
    - Gestión de mensajes de datos.
    - Gestión de la creación y actualización de los **token** de registro.
  - Escribir nuestra aplicación.

Prerequisitos de la plataforma FCM para Android

- Un dispositivo con Android 4.1 o posterior que también tengan la app Google Play Store instalada o un emulador que ejecute Android 4.1 cons las API de Google.
- El SDK de los servicios de Google Play proporcionado por el Android SDK Manager.
- La versión de Android Studio 1.5 o posterior.
- Un proyecto de Android Studio y su nombre de paquete.
- La compilación del proyecto debe ser para Android 4.1 (Jelly Bean) o superior.

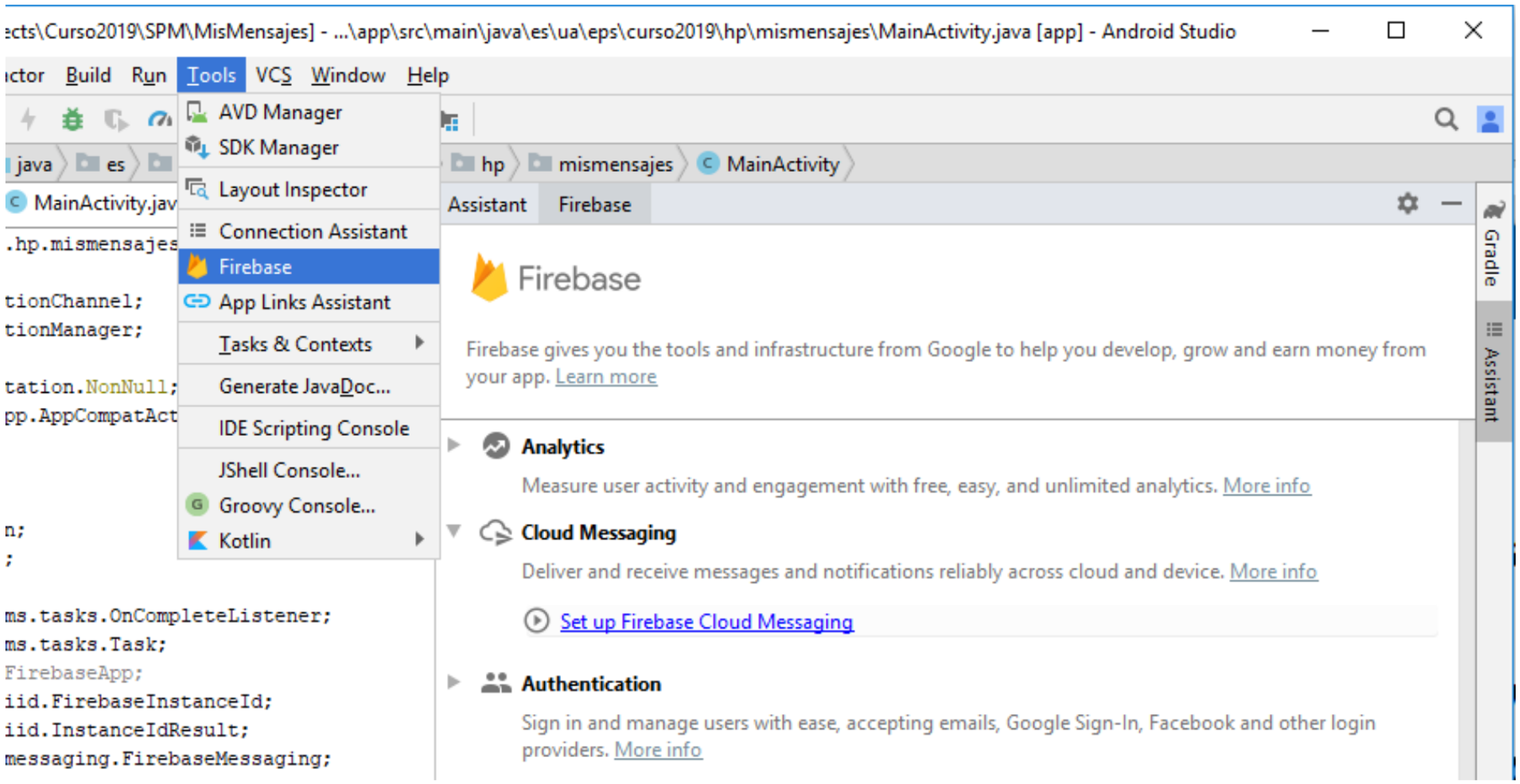
Habilitación de la API FCM

La habilitación de la API FCM la podemos realizar de dos formas distintas:

1. Si estamos usando la última versión de Android Studio (versión 2.2 o posterior), se puede realizar a través del **Asistente de Firebase**.
2. Si estamos usando una versión más antigua o se requiere una configuración de proyecto más compleja, podemos realizar una **configuración manual**.

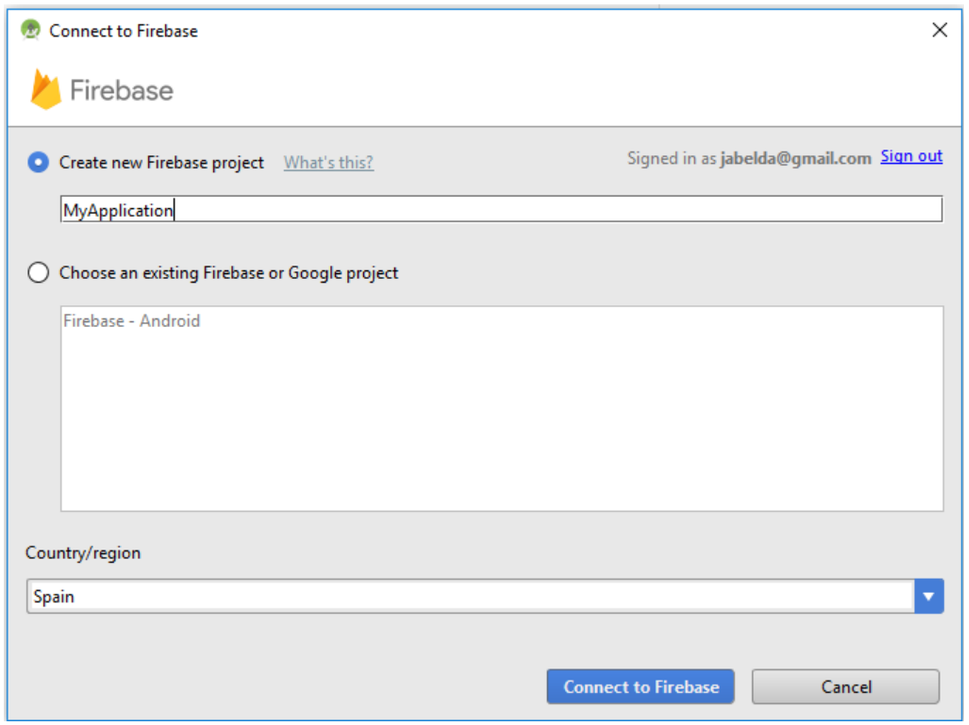
Usar el Asistente de Firebase

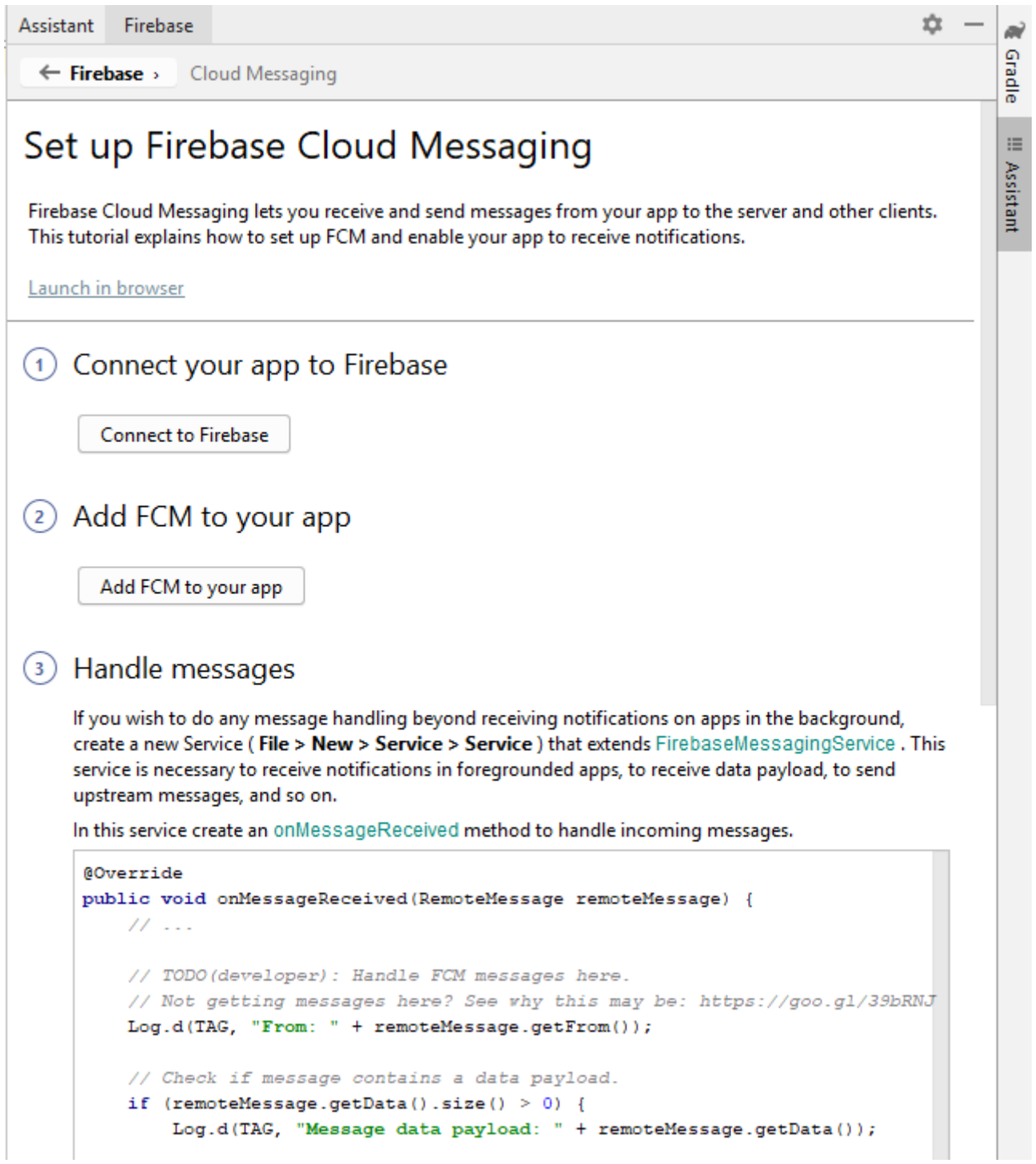
- En Android Studio hacer click en **Tools > Firebase** para abrir la Ventana del asistente.
- En la ventana del asistente, desplegar **Cloud Messaging** y seleccionar **Set up Firebase Cloud Messaging**.



Usar el Asistente de Firebase

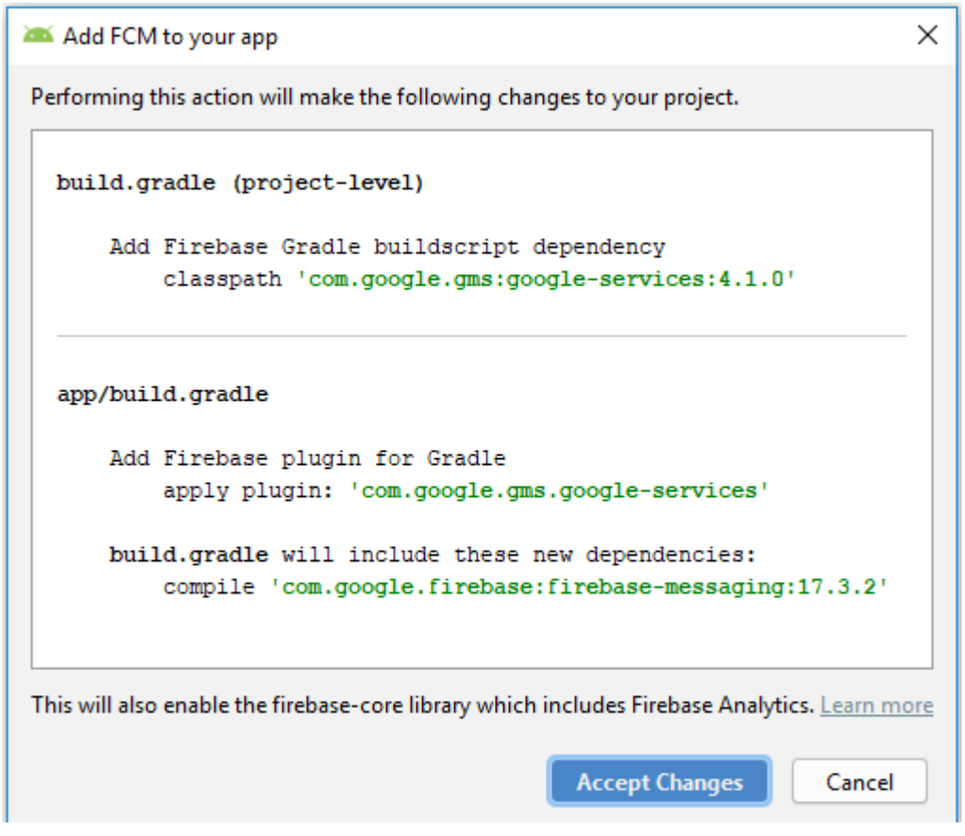
- Seguimos cada uno de los pasos del asistente
  - **Connect to Firebase**.
    Si es la primera vez que se hace en nuestra instalación de Android Studio, nos pide autorización para conectarse en nuestro nombre.

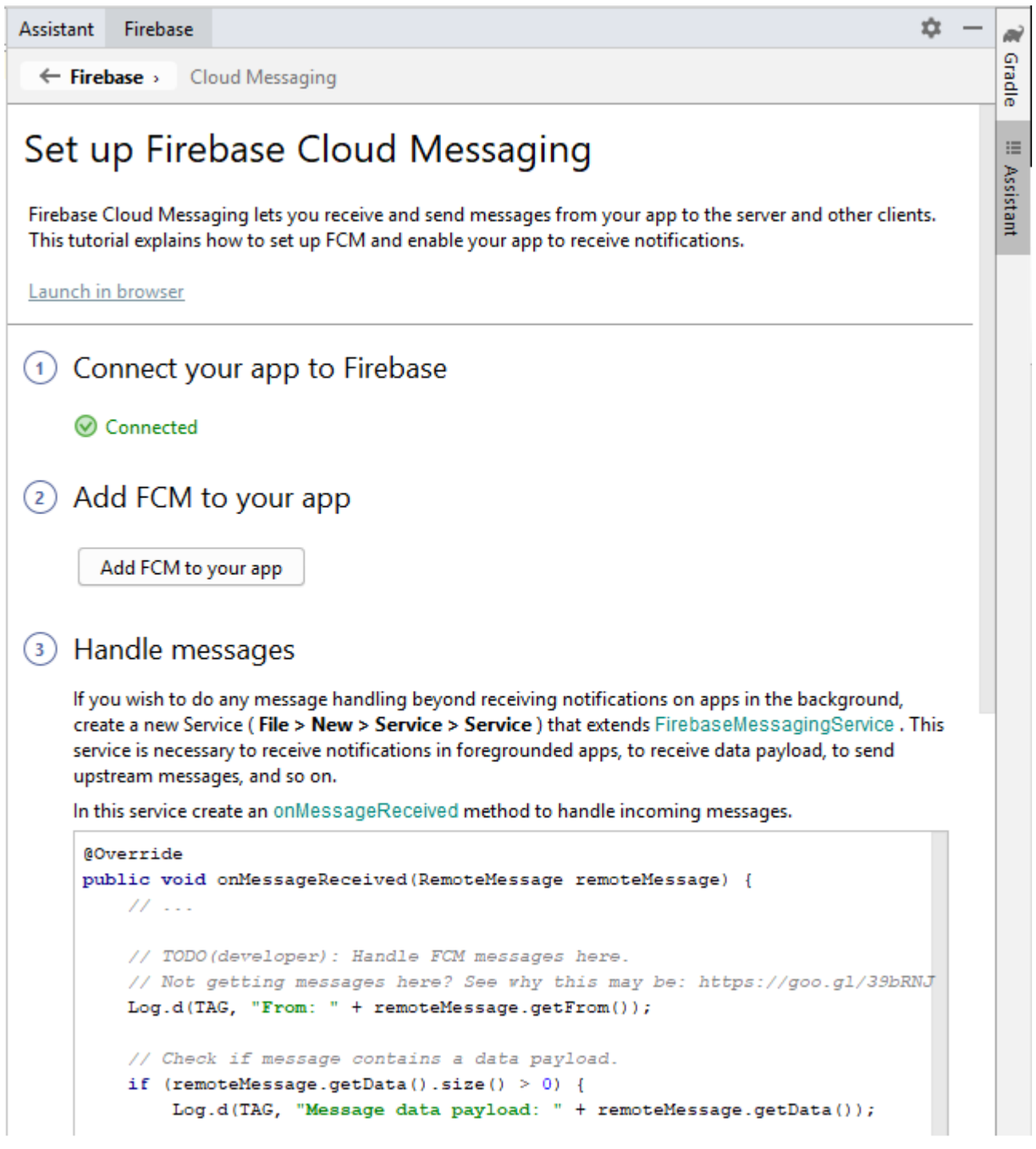




Usar el Asistente de Firebase

- Seguimos cada uno de los pasos del asistente.
  - **Add FCM to your app**. Aceptamos las modificaciones que nos propone.

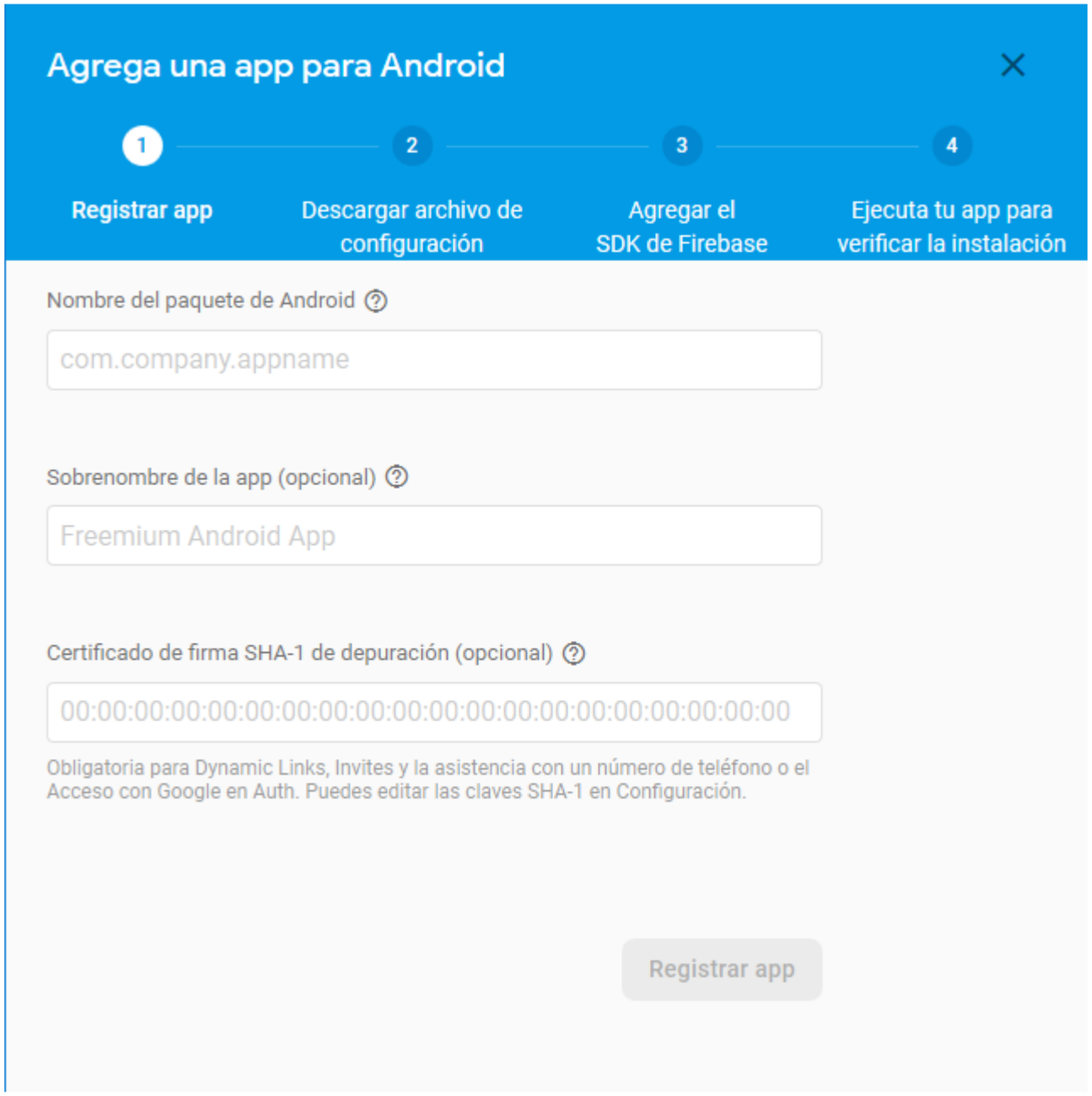




Habilitación de la API FCM manualmente

1. Ir a la consola de Firebase. https://console.firebase.google.com
2. Crear un nuevo proyecto o seleccionar uno existente.
3. Pulsar en Añadir aplicación y seguir las instrucciones.
4. Descargar el fichero **google-services.json**.
5. Copiarlo en la carpeta del módulo del proyecto, generalmente **app/**.

Añade Firebase a tu aplicación de Android



MAC/LINUX WINDOWS

```
$ move path-to-download/google-services.json app/
```

Habilitación de la API FCM manualmente

6. Añadir el plugin de google-services al fichero build.gradle del proyecto.

```
buildscript {
    // ...
    dependencies {
        // ...
        classpath 'com.google.gms:google-services:4.2.0' // google-services plugin
    }
}

allprojects {
    // ...
    repositories {
        // ...
        google() // Google's Maven repository
    }
}
```

7. Añadir el plugin de gradle así como las dependencias de los SDK de Firebase que queramos utilizar.

```
apply plugin: 'com.android.application'

android {
  // ...
}

dependencies {
  // ...
  implementation 'com.google.firebase:firebase-core:16.0.7'
  implementation 'com.google.firebase:firebase-messaging:17.3.4'

  // Getting a "Could not find" error? Make sure you have
  // added the Google maven respository to your root build.gradle
}

// ADD THIS AT THE BOTTOM
apply plugin: 'com.google.gms.google-services'
```

Configuración de los servicios a utilizar

- Un servicio que extiende de **FirebaseMessagingService**. Necesario para recibir notificaciones en primer plano, para recibir la carga de datos, acceder al refresco del token de registro del dispositivo, ...

```
<service android:name=".MyFirebaseMessagingService">
    <intent-filter>
        <action android:name="com.google.firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Definición del icono y color por defecto

- Android muestra el icono y el color predeterminado para:
  - Todos los mensajes de notificación enviados desde la consola de notificaciones.
  - Cualquier mensaje de notificación que no lo establezca de manera explícita.
- Si no se estable ningún icono predeterminado, y en el mensaje de notificación no se indica, se utilizará el icono de la aplicación mostrado en blanco.

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

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

AndroidManifest.xml

Recepción de mensajes

- Acceder al **token de registro**.
  - Se genera en el arranque inicial de la app.
  - Necesario para seleccionar como destino dispositivos individuales o crear grupos de dispositivos.
  - Se pasa como parámetro al método **onNewToken()**.
- El token de registro puede cambiar en las siguientes situaciones:
  - La app borra un ID de instancia.
  - La app se restablece en un dispositivo nuevo.
  - El usuario desinstala y vuelve a instalar la app.
  - El usuario borra los datos de la app.

Acceder al token de registro

```
/**
 * Called if InstanceID token is updated. This may occur if the security of
 * the previous token had been compromised. Note that this is called when the InstanceID token
 * is initially generated so this is where you would retrieve the token.
 */
@Override
public void onNewToken(String token) {
    Log.d(TAG, "Refreshed token: " + token);

    // If you want to send messages to this application instance or
    // manage this apps subscriptions on the server side, send the
    // Instance ID token to your app server.
    sendRegistrationToServer(token);
}
```

Recuperar el token de registro actual

```
FirebaseInstanceId.getInstance().getInstanceId()
        .addOnCompleteListener(new OnCompleteListener<InstanceIdResult>() {
            @Override
            public void onComplete(@NonNull Task<InstanceIdResult> task) {
                if (!task.isSuccessful()) {
                    Log.w(TAG, "getInstanceId failed", task.getException());
                    return;
                }

                // Get new Instance ID token
                String token = task.getResult().getToken();

                // Log and toast
                String msg = getString(R.string.msg_token_fmt, token);
                Log.d(TAG, msg);
                Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();
            }
        });
```

Suscripción a un tema

```
FirebaseMessaging.getInstance().subscribeToTopic("weather")
        .addOnCompleteListener(new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                String msg = getString(R.string.msg_subscribed);
                if (!task.isSuccessful()) {
                    msg = getString(R.string.msg_subscribe_failed);
                }
                Log.d(TAG, msg);
                Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();
            }
        });
```

Recepción de mensajes

- Sobrescribir el método **onMessageReceived**. Método que se lanza en la mayoría de mensajes a excepción de:
  - Mensajes de notificación o notificación y datos entregadas cuando la aplicación está en segundo plano.

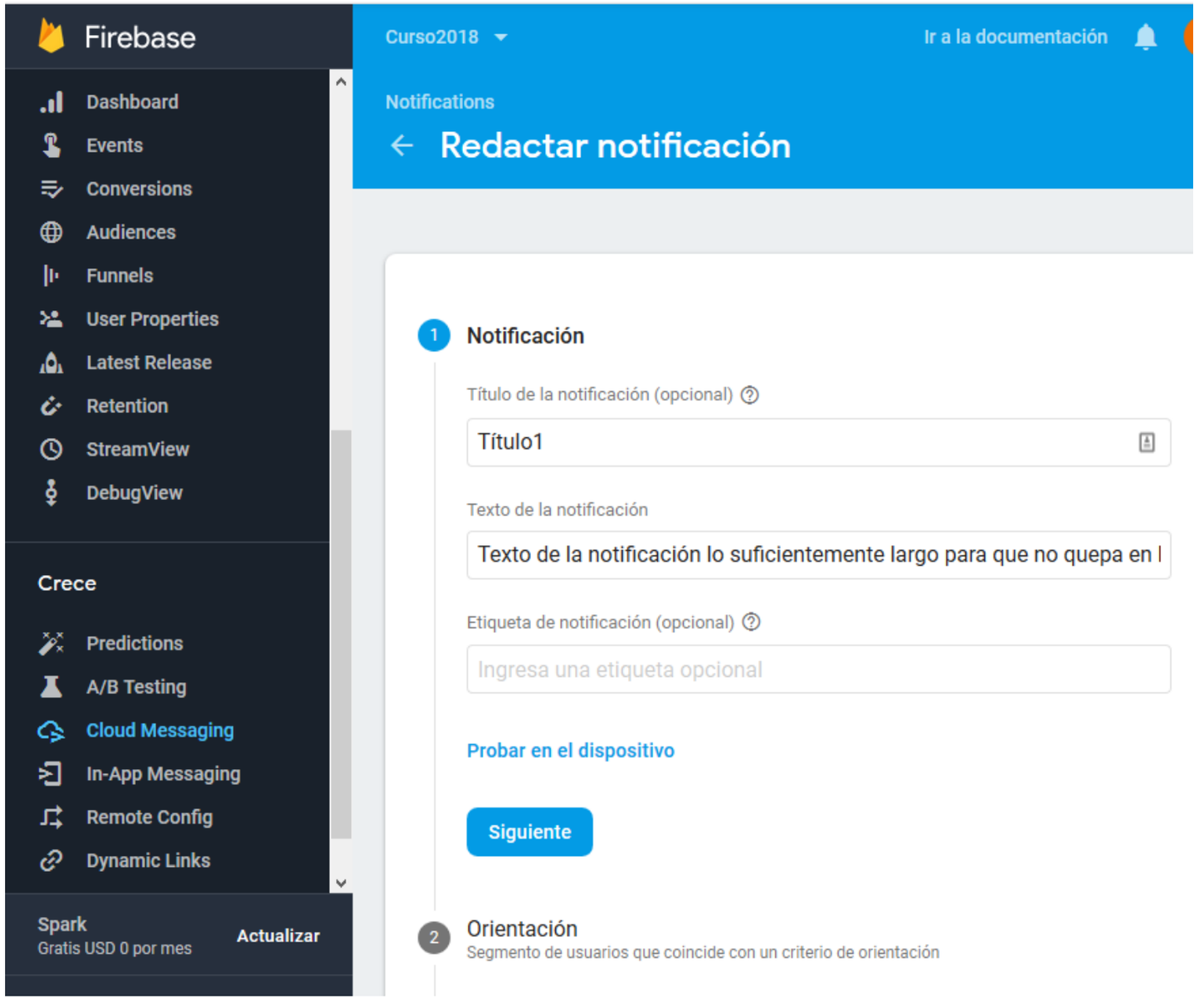
Estado de la app	Notificación	Datos	Ambos
Primer plano	`onMessageReceived`	`onMessageReceived`	`onMessageReceived`
Segundo plano	Bandeja del sistema	`onMessageReceived`	Notificación: Bandeja del sistema Datos: en extras de la intención.

Sobrescribir el método **onMessageReceived**

```java
@Override
public void onMessageReceived(RemoteMessage remoteMessage) {
    // ...

    // TODO(developer): Handle FCM messages here.
    // Not getting messages here? See why this may be: https://goo.gl/39bRNJ
    Log.d(TAG, "From: " + remoteMessage.getFrom());

    // Check if message contains a data payload.
    if (remoteMessage.getData().size() > 0) {
        Log.d(TAG, "Message data payload: " + remoteMessage.getData());

        if (/* Check if data needs to be processed by long running job */ true) {
            // For long-running tasks (10 seconds or more) use WorkManager.
            scheduleJob();
        } else {
            // Handle message within 10 seconds
            handleNow();
        }

    }

    // Check if message contains a notification payload.
    if (remoteMessage.getNotification() != null) {
        Log.d(TAG, "Message Notification Body: " + remoteMessage.getNotification().getBody());
    }

    // Also if you intend on generating your own notifications as a result of a received FCM
    // message, here is where that should be initiated. See sendNotification method below.
}
```

Envío de notificaciones a la aplicación

- Desde la consola de notificaciones.
- Desde el servidor de aplicaciones.




```
https://fcm.googleapis.com/fcm/send
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA

{ "notification": {
    "title": "España vs Italia",
    "body": "2 - 0"
  },
  "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1..."
}
```

```
https://gcm-http.googleapis.com/gcm/send
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA

{ "data": {
    "message": "Contenido del mensaje"
  },
  "to" : "bk3RNwTe3H0:CI2k_HHwgIpoDKCIZvvDMExUdFQ3P1..."
}
```

Envío de mensajes a temas

- A un único tema

```
https://fcm.googleapis.com/fcm/send
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA
{
  "to": "/topics/foo-bar",
  "data": {
    "message": "This is a Firebase Cloud Messaging Topic Message!",
   }
}
```

- A varios temas

```
https://fcm.googleapis.com/fcm/send
Content-Type:application/json
Authorization:key=AIzaSyZ-1u...0GBYzPu7Udno5aA
{
  "condition": "'dogs' in topics || 'cats' in topics",
  "data": {
    "message": "This is a Firebase Cloud Messaging Topic Message!",
   }
}
```

Criterios selección servidor FCM HTTP/XMPP

- Mensajes de Subida/Bajada
  - HTTP: De bajada solamente: nube-a-dispositivo.
  - XMPP: De subida y de bajada (dispositivo-a-nube, nube-a-dispositivo)
- Mensajes (síncronos o asíncronos)
  - HTTP: Las aplicaciones de servidor envían mensajes mediante peticiones HTTP POST y esperan una respuesta. Este mecanismo es sincrónico y hace que el remitente se bloquee antes de enviar otro mensaje.
  - XMPP: Las aplicaciones de servidor se conectan a la infraestructura de Google mediante una conexión XMPP persistente y envían/reciben mensajes a/desde todos sus dispositivos a la velocidad que de la línea. XMPP envía una notificación de acuse de recibo o de fallo (en forma de mensajes XMPP ACK y NACK especiales codificados en JSON) de forma asíncrona.
- JSON
  - HTTP: Los mensajes JSON se envían como POST por HTTP.
  - XMPP: Los mensajes JSON se encapsulan en los mensajes XMPP .
- Texto Plano
  - HTTP: Los mensajes de texto plano se envían como POST por HTTP.
  - XMPP: No los soporta.
- Envío a múltiples ID de registro.
  - HTTP: Soportado en los mensajes con formato JSON.
  - XMPP: No los soporta.

Funcionamiento Aplicación de Servidor

1. La aplicación de servidor envía un mensaje a los servidores de FCM.
2. Google encola y almacena el mensaje en caso de que el dispositivo esté desconectado.
3. Cuando el dispositivo está conectado, Google envía el mensaje al dispositivo.
4. Si la aplicación está en segundo plano, la notificación se entrega a la bandeja del sistema.
5. Si se recibe un mensaje de datos, o la aplicación está en primer plano, esta procesa el mensaje.

Requisitos para el envío de mensajes

- Destino

  Obligatorio. Cuando la aplicación de servidor envía un mensaje en FCM, debe especificar un destino.

  Para HTTP debemos especificar el destino de una de las siguientes formas:

  - to: El valor debe ser un Token de registro, una clave de notificación o un tema.
  - registration_ids: Para enviar a 1 o más dispositivos (hasta a 1000). Cuando se envía un mensaje a varios ID de registro, se llama un mensaje de multidifusión.
  - notification_key: Está desaprobado. En su lugar utilizar to.
  - condition: Expresión lógica de las condiciones que determinan el destinatario del mensaje

  Para XMPP:

  - to: Puede contener un único Token de registro o una clave de notificación. XMPP no admite mensajes de multidifusión.
  - condition: Expresión lógica de las condiciones que determinan el destinatario del mensaje
- Contenido

  Opcional. Si vamos a incluir un contenido en el mensaje, se utiliza el parámetro data, el notification o ambos. Esto se aplica tanto para HTTP como para XMPP.
- Parámetros del mensaje

  Varían en función del servidor de conexión y el formato del mensaje.

Opciones de los mensajes HTTP (JSON)

Parámetro	Descripción
collapse_key	Este parámetro especifica una cadena arbitraria que se utiliza para contraer un grupo de mensajes cuando el dispositivo no está disponible, de modo que sólo el último mensaje se envía al cliente. **Opcional**.
priority	Establece la prioridad del mensaje. Los valores válidos son “normal” y “high”. De forma predeterminada, los mensajes de notificación se envían con prioridad alta y los mensajes de datos se envían con prioridad normal. La prioridad normal optimiza el consumo de la batería de la app cliente y debe usarse salvo que el mensaje deba entregarse inmediatamente. En el caso de los mensajes con prioridad normal, la app puede recibir los mensajes con una demora no específica.
content_available	Cuando se envía un mensaje o una notificación con este parámetro a true, se despierta la aplicación cliente si está inactiva. **Opcional.**
time_to_live	Este parámetro especifica el tiempo (en segundos) que el mensaje se debe conservar en el almacenamiento de FCM si el dispositivo se encuentra sin conexión. El tiempo máximo admitido es de 4 semanas, al igual que el valor predeterminado.
restricted_package_name	Este parámetro especifica el nombre del paquete de la aplicación donde los token de registro deben coincidir para recibir el mensaje.
dry_run	Este parámetro permite a los desarrolladores probar una solicitud sin tener que enviar un mensaje. El valor predeterminado es **false**.

Parámetros en los mensajes de texto sin formato

Parámetro	**Descripción**
registration_id	Este parámetro especifica el ID de registro del único dispositivo que recibe el mensaje. **Obligatorio**.
collapse_key	Igual que JSON (ver tabla anterior). **Opcional**.
data.<key>	Este parámetro especifica los datos de carga útil, expresadas como parámetros prefijados con data. y el sufijo como clave. **Opcional**.
delay_while_idle	Este parámetro especifica si los mensajes deben ser entregados cuando el dispositivo está dormido. Un valor de 1 o true indica cierto, y cualquier otra cosa indica falsa. **Opcional**. El valor predeterminado es false.
time_to_live	Igual que JSON (ver tabla anterior). **Opcional**.
restricted_package_name	Igual que JSON (ver tabla anterior). **Opcional**.
dry_run	Igual que JSON (ver tabla anterior). **Opcional**.

Soporte de carga de notificación en Android

Parámetro	Descripción
title	Indica título de notificación.
body	Indica texto del cuerpo de la notificación.
android_channel_id	El ID de canal de la notificación (nuevo en Android O).
icon	Indica el ícono de notificación.
sound	Indica un sonido para reproducir cuando el dispositivo recibe la notificación. Admite default o el nombre de archivo de un recurso de sonido contenido en la app. Los archivos de sonido de Android deben residir en /res/raw/.
tag	Indica si cada mensaje de notificación genera una nueva entrada en el panel de notificaciones. Si no se configura, cada solicitud crea una nueva notificación. Si se configura y ya hay una notificación con la misma etiqueta, la nueva notificación reemplaza la existente en el panel lateral de notificaciones.
color	Indica el color del ícono, expresado en formato #rrggbb
click_action	Indica la acción asociada con el clic de un usuario en la notificación.
body_loc_key	Indica la clave en la string de cuerpo para la localización.
body_loc_args	Indica el valor de string para reemplazar especificadores de formato en string de cuerpo para la localización.
title_loc_key	Indica la clave en la string de título para la localización.
title_loc_args	Indica el valor de string para reemplazar especificadores de formato en string de título para la localización.

¿Preguntas...?