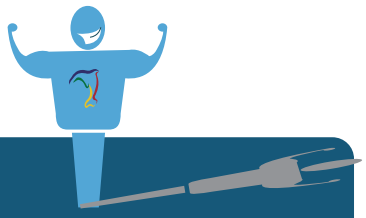
autentia

Avenida de Castilla,1 - Edificio Best Point - Oficina 21B
28830 San Fernando de Henares (Madrid)
tel./fax: +34 91 675 33 06
info@autentia.com - **www.autentia.com**

¿**Qué ofrece** Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
Ese apoyo que siempre quiso tener...

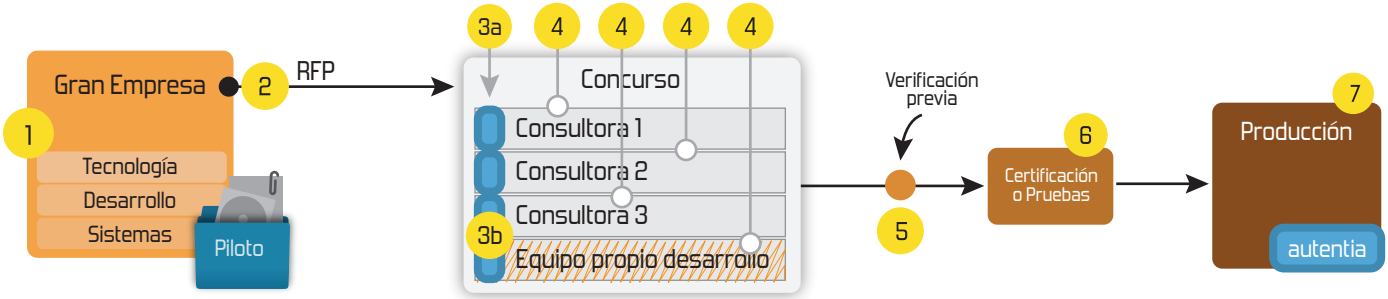
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

- Spring MVC, JSF-PrimeFaces /RichFaces, HTML5, CSS3, JavaScript-jQuery
- Gestor portales (Liferay)
- Gestor de contenidos (Alfresco)
- Aplicaciones híbridas
- Control de autenticación y acceso (Spring Security)
- UDDI
- Web Services
- Rest Services
- Social SSO
- SSO (Cas)
- JPA-Hibernate, MyBatis
- Motor de búsqueda empresarial (Solr)
- ETL (Talend)
- Dirección de Proyectos Informáticos.
- Metodologías ágiles
- Patrones de diseño
- TDD
- Tareas programadas (Quartz)
- Gestor documental (Alfresco)
- Inversión de control (Spring)
- BPM (jBPM o Bonita)
- Generación de informes (JasperReport)
- ESB (Open ESB)

Compartimos nuestro conociemiento en:
www.adictosaltrabajo.com

Para más información visítenos en:
www.autentia.com

Miguel Arlandy Rodríguez

Consultor tecnológico de desarrollo de proyectos informáticos.

Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Somos expertos en Java/JEE

Ver todos los tutoriales del autor

Fecha de publicación del tutorial: 2014-07-15

Tutorial visitado 1 veces Descargar en PDF

Configurando Notificaciones Push para desarrollos Android con Google Cloud Messaging.

0. Índice de contenidos.

1. Introducción

En términos de tecnología de movilidad, las **Notificaciones Push son aquellos mensajes que recibimos en el dispositivo** y que han sido **emitidos desde cualquier punto de un sistema**. Tenemos un ejemplo muy claro en el popular WhatsApp, donde son los usuarios los que envían mensajes a los dispositivos de otros usuarios. Otro ejemplo con notificaciones enviadas de manera automática (y no manual como WhatsApp) podría ser una aplicación de cliente de correo, cuando el servidor detecta un mensaje entrante envía una notificación al dispositivo del usuario.

Las Notificaciones Push permiten el **envío de mensajes** desde cualquier parte de un sistema a una aplicación móvil tanto si la aplicación está siendo utilizada por el usuario, si está corriendo en un segundo plano, si todavía no ha sido arrancada o, incluso, si el dispositivo está en reposo.

En este tutorial intentaremos explicar cómo poder recibir Notificaciones Push en nuestra aplicación Android haciendo uso del servicio de mensajería en la nube: Google Cloud Messaging.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil MacBook Pro 15' (2.2 Ghz Intel Core I7, 8GB DDR3).
- Sistema Operativo: Mac OS X Mavericks 10.9

3. ¿Cómo funcionan las notificaciones con Google Cloud Messaging?.

Las notificaciones Push en Android mediante Google Cloud Messaging (GCM) se ejecutan en un escenario que está compuesto por, al menos, tres actores:

- **Google Cloud Messaging**: El servicio de Google habilitado para el envío de Notificaciones Push a dispositivos Android.
- **Servidor**: con un servicio (REST, SOAP, aplicación web, etc...) que será el encargado de gestionar los

identificadores de registro de dispositivos a los que podemos enviar las notificaciones y de comunicarse con GCM solicitando el envío de notificaciones al dispositivo (o dispositivos) deseado.

- **Dispositivo Android**: que recibirá las notificaciones.

Para poder enviar notificaciones a un dispositivo Android desde GCM dicho dispositivo debe estar antes registrado en el servicio.

3.1 El registro del dispositivo.

Lo primero que debemos hacer para que nuestra aplicación Android pueda recibir Notificaciones Push desde Google Cloud Messaging (GCM) será registrarla en dicho servicio. Es una forma de decirle a GCM: *"soy un dispositivo que quiere recibir notificaciones de una aplicación"*. La forma que tenemos de decirle la aplicación de la que queremos recibir notificaciones es indicándole un **número de proyecto** (lo veremos en el siguiente apartado). Esto sería lo equivalente al **paso 1** del diagrama que viene a continuación.

Si todo está correcto, GCM nos responderá con un identificador de registro. Algo del tipo:

```
APA33bH6YRxig6cFFUE_utY2aEaEhVThPDkh5xQ6pFbjf-GKkSLZceGkCy8wjhx8QxdfgpMbFSZnAqPcJ5hmDhgkK
```

Esto se correspondería con el **paso 2** del diagrama.

¿Y qué hacemos con ese identificador de registro?. Pues ahí es donde entra en juego nuestro nuestro servidor, en concreto el servicio que tenemos desplegado en él.

Lo que haremos será **enviar ese identificador a nuestro servicio** (vía HTTP, por ejemplo...) de forma que éste lo pueda almacenar ya que lo necesitará cuando se comunique con GCM para indicarle que debe enviar una notificación (lo veremos en el siguiente punto). Por supuesto, esa petición de envío del identificador de registro **la podemos complementar con cualquier otra información adicional** como el usuario del dispositivo, caracterísiticas o lo que sea ya que el servicio es nuestro. Esto sería lo relativo al **paso 3** del diagrama.

3.2 El envío de la notificación.

Podemos enviar una Notificación Push a cualquiera de los dispositivos Android que tengamos registrados en nuestro servicio desde **cualquier parte del sistema**.

Para ello, nuestro servicio tendrá una operación donde recibirá la **información relativa al mensaje** que queremos enviar en forma de notificación y **el destinatario** o destinatarios. La información relativa al destinatario puede ser directamente el identificador del registro u otra información que el servicio sepa relacionar con dicho identificador de registro. Esto sería lo equivalente al **paso 1** del siguiente diagrama.

Últimos Tutoriales del Autor

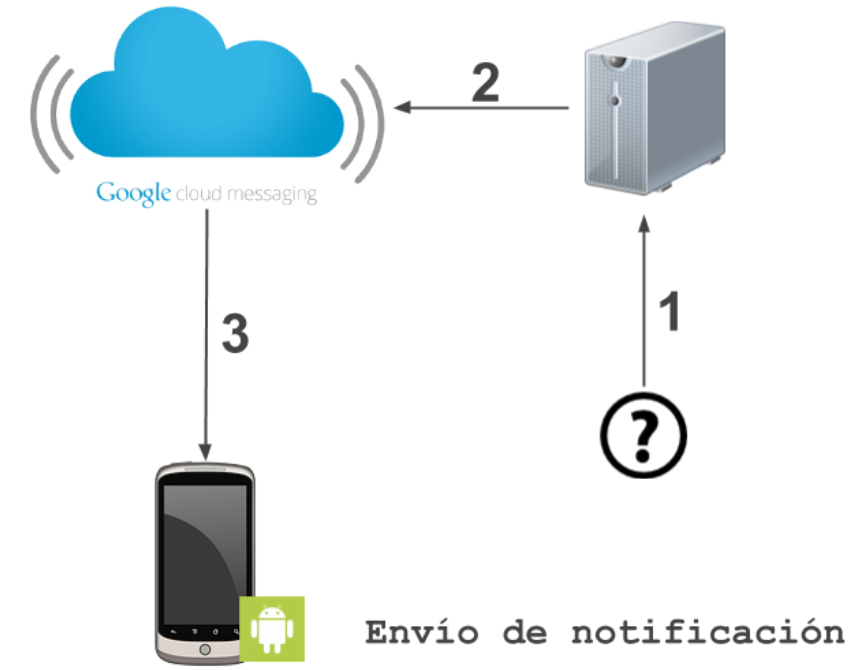
» Introducción a WSO2 API Manager

» SOA vs. SOAP y REST

» REST, el principio HATEOAS y Spring HATEOAS

» SOA y los tipos de servicios

» Introducción a Spring Batch



A continuación, con la información de la petición de envío de notificación que recibió nuestro servicio, éste envía una nueva petición a GCM (puede hacerse de manera síncrona o asíncrona: HTTP o XMPP) para que mande la notificación al dispositivo. El destinatario de la notificación se indica mediante el **identificador de registro** que obtuvimos en el punto anterior. Necesitaremos adjuntar unas **credenciales de servidor** a nuestra petición. En el punto siguiente veremos cómo obtenerlas. Esto sería lo equivalente al **punto 2** del diagrama.

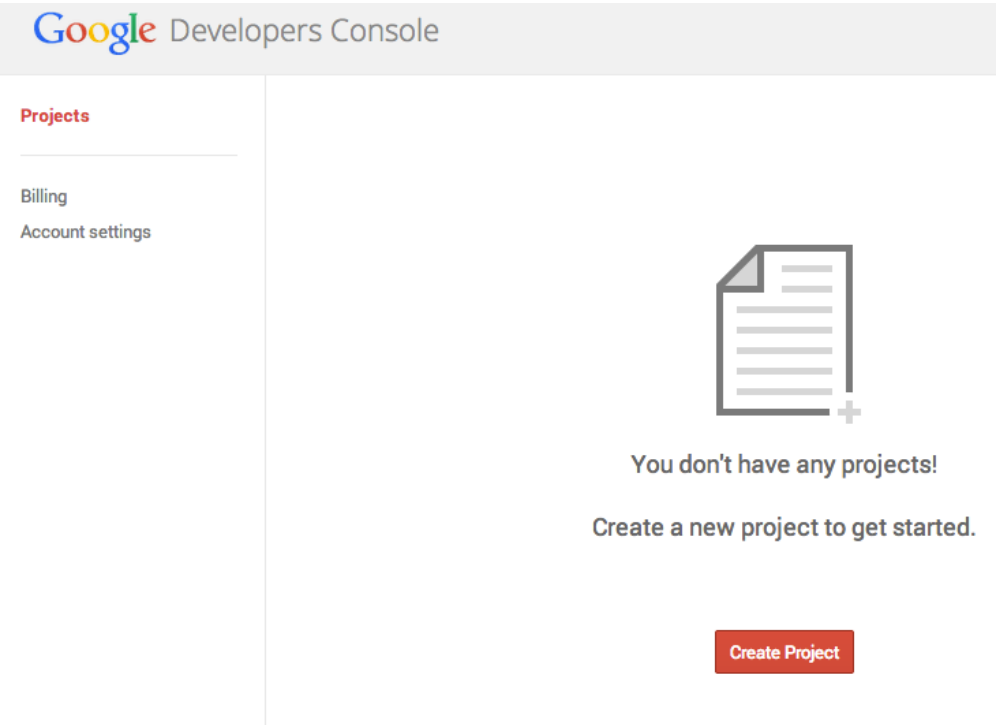
Una vez que hemos enviado la petición a Google Cloud Messaging con **nuestras credenciales del servidor**, la información relativa a la **notificación** y su **destinatario**, GCM enviará dicha Notificación Push. **Paso 3** del diagrama.

4. Habilitando el servicio Google Cloud Messaging.

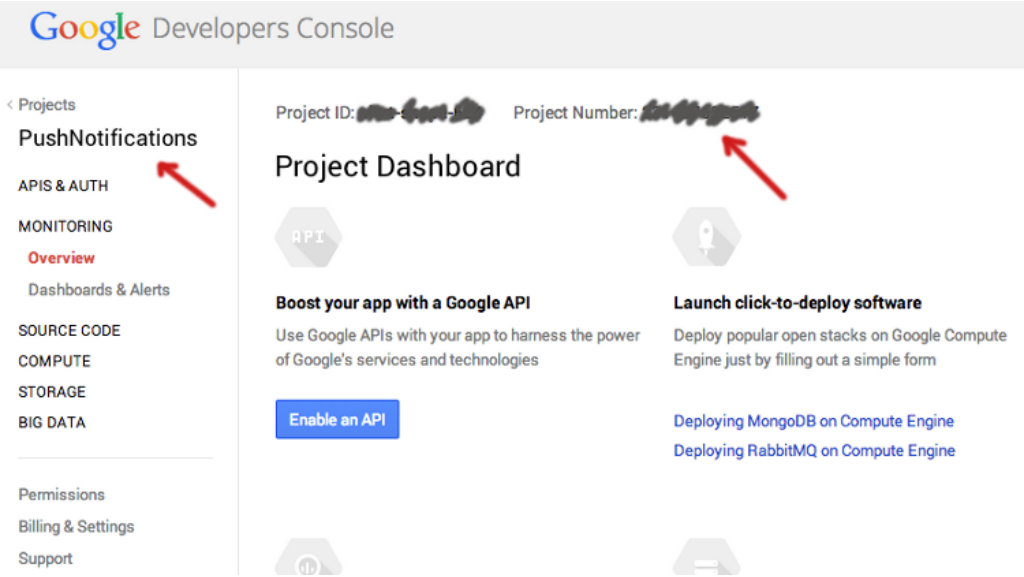
Para habilitar el servicio Google Cloud Messaging debemos seguir una serie de sencillos pasos:

4.1 Creando un proyecto Google API.

Lo primero que haremos será crear un proyecto en Google API, para ello accedemos a la consola de desarrolladores de Google con nuestro usuario.



En la sección "Projects" pulsamos sobre el botón "Create Project", le asignamos un nombre y ya tendremos nuestro proyecto creado. Nos aparecerá una pantalla donde podremos ver información relevante a dicho proyecto.

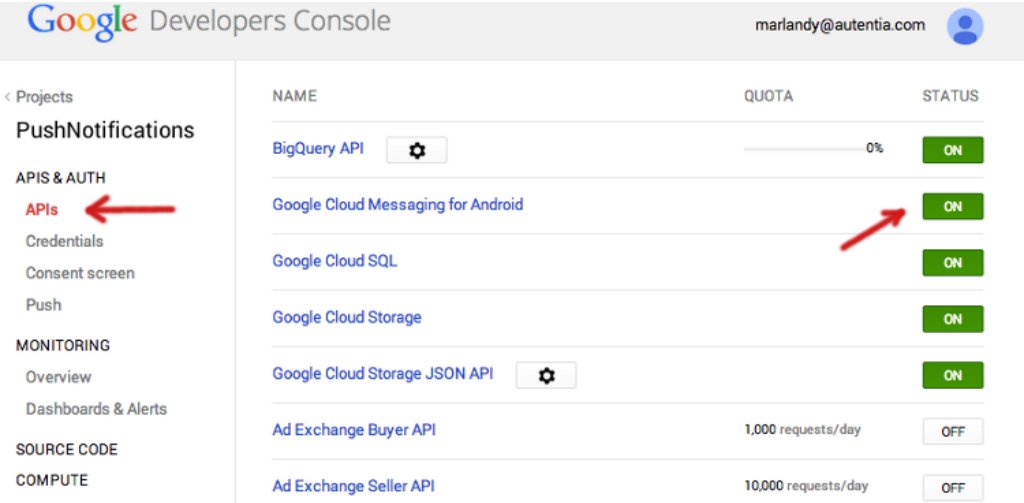


Para las Notificaciones Push, el campo que más nos interesa será **Project Number**. Dicho número es necesario para el paso de registro del dispositivo en GCM tal y como vimos en el punto 3.1. Dicho **Project Number** será algo del tipo:

```
231447612989
```

4.2 Habilitando la opción Google Cloud Messaging.

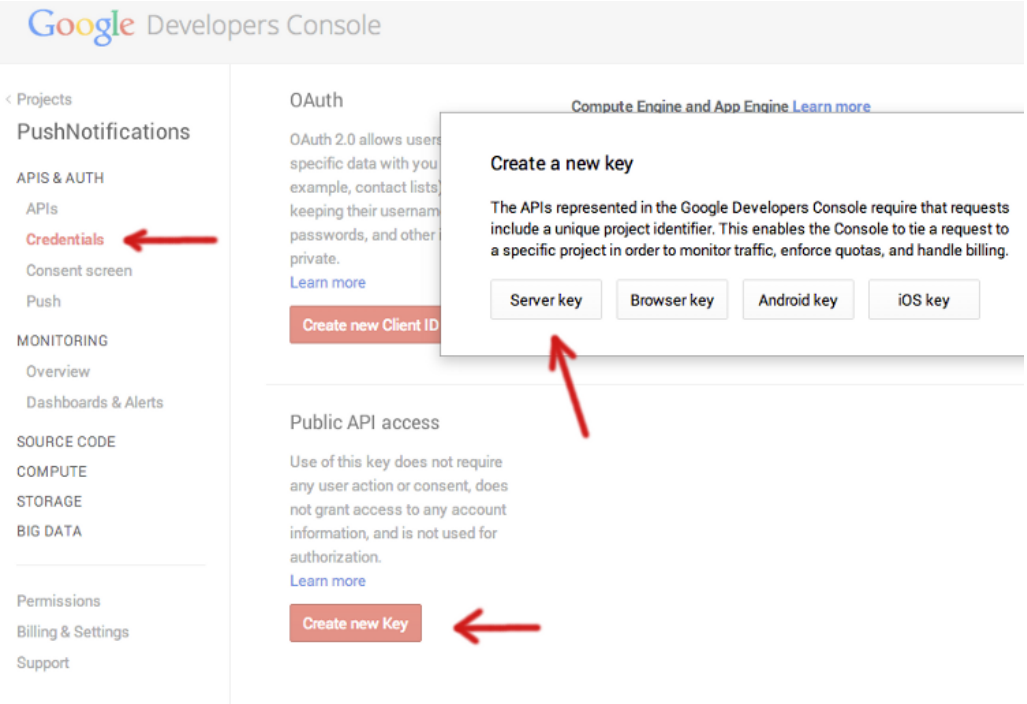
Una vez tenemos el proyecto creado, lo siguiente que haremos será habilitar el servicio de Google Cloud Messaging. Para ello, pulsamos en el menú de la izquierda sobre APIS & AUTH > APIs y activamos el servicio "Google Cloud Messaging for Android" (por defecto estará desactivo).



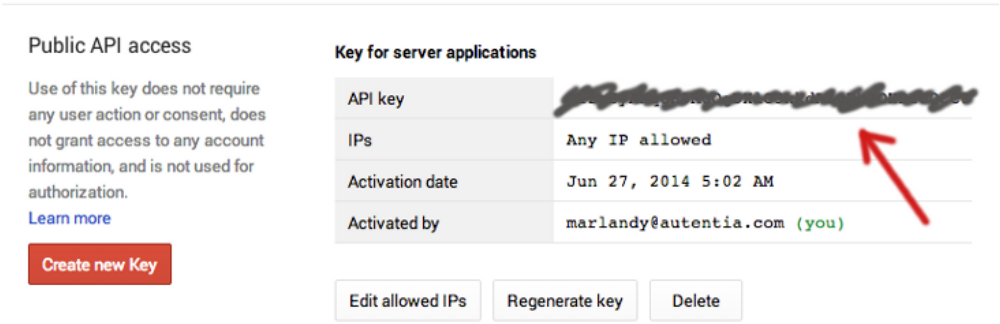
4.3 Obteniendo la clave de acceso al API.

Por último, necesitaremos tener una clave de acceso a nuestro API para que podamos enviar peticiones a GCM y que éste las sirva las notificaciones al dispositivo o dispositivos destino. Esta clave de acceso será utilizada por nuestro servicio tal y cómo vimos en el **paso 2** del envío de la notificación.

Para generar la clave de acceso que usará nuestro servidor pulsamos en **APIS & AUTH** > **Credentials** y en la sección **Public API Access**, pulsamos sobre el botón **"Create New Key"**. En el diálogo que nos aparecerá, pulsamos en **"Server Key"**.



Y con esto obtendremos la clave de acceso a nuestra API que será utilizada por nuestro servicio.



Nuestra **API key** sería algo del tipo:

```
AIveCyANz3H5MuOgDWrC6xkdECqRbIOHBzOQDC6
```

5. Nuestro servicio.

Como vimos anteriormente, además del servicio en la nube Google Cloud Messaging necesitaremos nuestro propio servicio que se encargue de, al menos, **registrar los identificadores** de registro que nos devuelve GCM y de **enviar las notificaciones** a los dispositivos.

Hemos creado un sencillo servicio REST en Java con Spring MVC que se ocupa de ambas cosas. Podéis encontrar el código fuente aquí.

5.1 Registrando los identificadores de registro de GCM.

La primera de las dos cosas que, al menos, debe realizar nuestro servicio es registrar los identificadores de registro que devuelve GCM cuando un dispositivo quiere recibir notificaciones. Mediante el identificador de registro le comunicamos a GCM a qué dispositivo queremos que se envíe la notificación.

Nuestro dispositivo móvil, una vez haya recibido el identificador de registro de GCM deberá enviar una petición a nuestro servicio con dicho identificador para que quede registrado (ver registro del dispositivo). La petición sería algo como:

```
Url: http://localhost:8080/gcm-rest/api/registrations
Content-Type: application/x-www-form-urlencoded
Method: POST
Params: registrationId=APA33bE8f_SCHrrUvTlfibvAyfpk3Ai9YoEIAPhn74tVkryBLolM0RHdbh53tC27VdR
```

5.2 Enviando las notificaciones.

Este punto se correspondería con el **paso 1** del proceso de envío de la notificación

Si queremos enviar una notificación a uno o varios de nuestros disposivitos **previamente registrados** deberemos hacer una petición indicando diferentes campos (ver envío de la notificación):

- **badge**: número mayor que 0 que aparecerá en el texto de la notificación. Indica el número de notificaciones que el usuario tiene pendientes de leer.
- **title**: el título del mensaje de nuestra notificación.
- **message**: el cuerpo del mensaje.
- **registrationIdsToSend**: listado de identificadores de registro de dispositivos a los que queremos que llegue la notificación.

La petición de envío a de notificación a uno o varios dispositivos sería algo como (acepta también formato XML):

```
Url: http://localhost:8080/gcm-rest/api/notifications
Content-Type: application/json
Method: POST
Request Raw Body:
{
    "badge":1,
    "title":"Notification title",
    "message":"A message",
    "registrationIdsToSend":[
    "APA91bE9f_SCHrrUvTlkibvAyfpk3Ai9YoEIAPhn50tVkryBLolM0RHdbh53tC27VdRcMTWwyervn4zL4SiD
    "OAN64bE7f_SCHrrUvTlkibvAyfpk9Ai3YoEIAPhn49tVkryaaOlC3LAdbh33tC42VdRcMTWwyervn5zL6SiD
    ]
}
```

Aunque no nos vamos a meter en detalles (os dejo el código fuente del servicio), cabe destacar que Google nos proporciona una librería en Java para poder conectar desde nuestro servidor con Google Cloud Messagging. Podéis añadir la siguiente dependencia en vuestro proyecto Maven.

```
<dependencies>
    <dependency>
        <groupid>com.google.android.gcm</groupid>
        <artifactid>gcm-server</artifactid>
        <version>1.0.2</version>
    </dependency>
</dependencies>

<repositories>
    <!-- GCM server repository -->
    <repository>
        <id>gcm-server-repository</id>
        <url>https://raw.github.com/slorber/gcm-server-repository/master/releases</url>
    </repository>
</repositories>
```

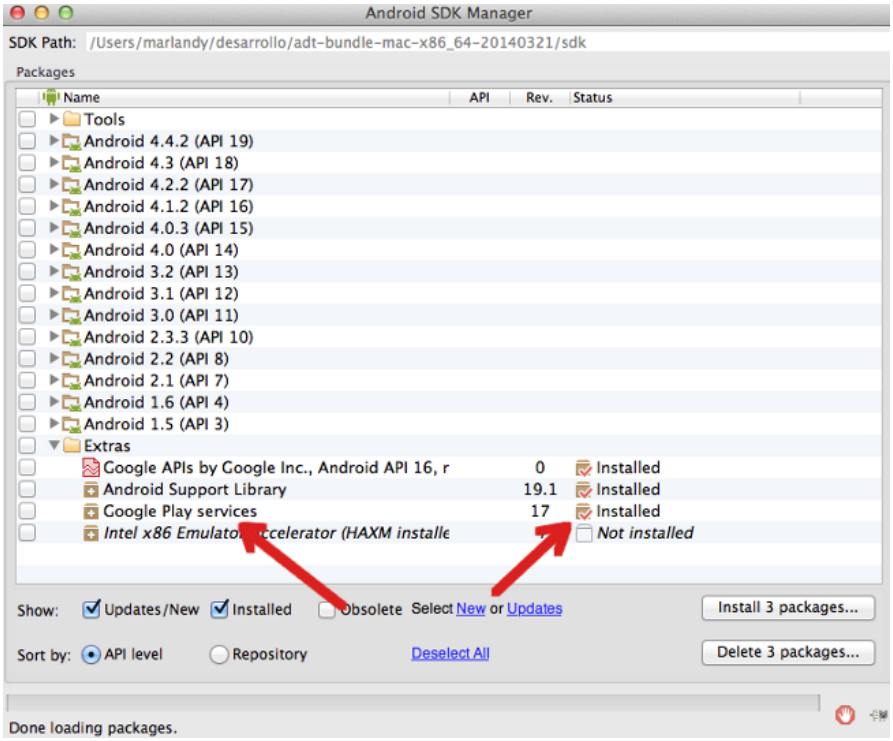
6. Condiciones que debe cumplir nuestra aplicación.

Para que nuestra aplicación Android pueda recibir notificaciones Push, debe cumplir con dos condiciones:

- Tener configurados los Google Play Services
- Tener habilitados una serie de permisos

6.1 Configurar Google Play Services.

Antes de habilitar Google Cloud Messaging debemos asegurarnos de que instalados los Google Play Services en nuestra SDK. Para ello, en nuestro Android SDK Manager, en la sección "Extras" podemos instalarlos en el caso de que lo hubiéramos hecho ya.



Para poder probar el proyecto necesitaremos un dispositivo Android con **versión 2.3 o superior** y **Google Play Store** instalado. Si deseamos probarlo en un **emulador**, podremos hacerlo con nuestro dispositivo virtual (AVD) con **Android 4.2.2 o superior**.

6.2 Habilitar los permisos necesarios.

Además de los servicios de Google Play, nuestra aplicación necesitará una serie de permisos para poder enviar y recibir notificaciones. Tales permisos se configuran en el fichero **AndroidManifest.xml** y son los siguientes:

- **com.google.android.c2dm.permission.RECEIVE**: permiso para que la aplicación se pueda registrar en GCM y recibir mensajes.
- **android.permission.INTERNET**: aunque no es obligatorio si que es muy recomendable. Gracias a este permiso podremos enviar el identificador de registro que hemos obtenido de GCM a nuestro servidor.
- **android.permission.GET_ACCOUNTS**: este permiso es necesario para versiones de Android inferiores a 4.0.4. Es necesario ya que GCM requiere una cuenta de Google.
- **android.permission.WAKE_LOCK**: permiso para mantener la pantalla apagada cuando llega una notificación.
- ***<paquete de la aplicación>*.permission.C2D_MESSAGE**: permiso para que únicamente esta aplicación pueda registrarse en GCM y recibir mensajes.

Además de habilitar los permisos citados anteriormente, necesitaremos:

- **Un receptor (receiver)**: con el siguiente permiso habilitado **com.google.android.c2dm.permission.SEND** de forma que GCM pueda enviarle mensajes y con el que podremos tratarlos.
- **Un servicio**: al que el receptor le envía el mensaje que recibió mientras se ocupa de que el dispositivo no se quede dormido en el proceso.

Nótese que, si para desarrollar nuestra aplicación utilizamos un **framework cross-platform** tipo Phonegap (Cordova) o Titanium, este proceso **nos lo podríamos ahorrar** mediante la configuración automática del plugin de notificaciones de cada uno de estos frameworks. Si el desarrollo es nativo debemos hacerlo manualmente.

7. Referencias.

- Código fuente del servicio.
- Google Play Messaging for Android
- Google Developers Console
- Implementing GCM Client
- Setting Up Google Play Services

8. Conclusiones.

En este tutorial hemos visto cómo habilitar las Notificaciones Push para que puedan ser enviadas a nuestra aplicación Android haciendo uso de Google Cloud Messaging. Las notificaciones son una característica muy interesante que nos proporcionan los dispositivos móviles y cuyo uso puede estar especialmente indicado en aplicaciones como: **chats, herramientas de trabajo colaborativo, sistemas de alertas y monitorización**, etc...

En próximos tutoriales veremos **cómo desarrollar una aplicación que reciba Notificaciones Push** desde Google Cloud Messaging en un dispositivo Android.

Cualquier duda en la sección de comentarios.

Espero que este tutorial os haya sido de ayuda. Un saludo.

Miguel Arlandy

marlandy@autentia.com

Twitter: @m_arlandy

A continuación puedes evaluarlo:

Regístrate para evaluarlo

Por favor, vota +1 o compártelo si te pareció interesante

Share | 8+1 0

Anímate y coméntanos lo que pienses sobre este **TUTORIAL:**

» **Registrate** y accede a esta y otras ventajas «

IMPULSA Impulsores Comunidad ¿Ayuda?

---
sin clicks

0 personas han traído clicks a esta página

+ + + + + + + +

powered by karmacracy

W3C XHTML 1.0 W3C CSS XML RSS XML ATOM