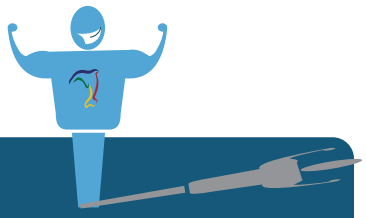
autentia

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...

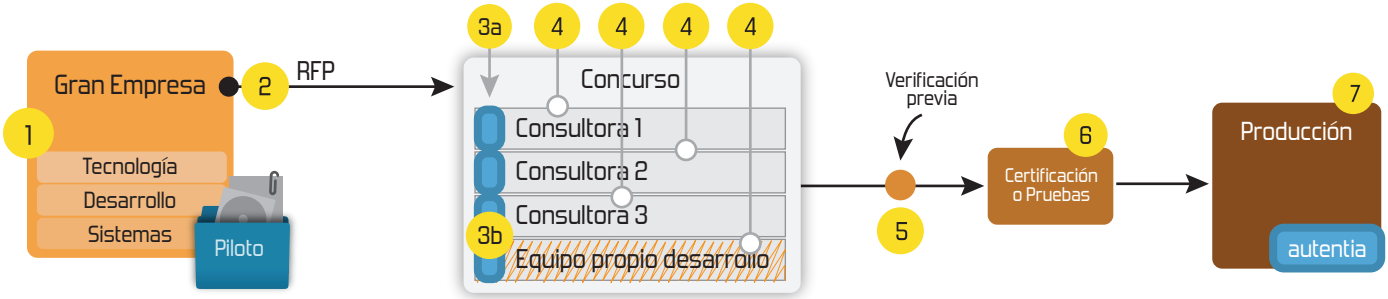
1. Desarrollo de componentes y proyectos a medida



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 conocimiento 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: 2014-07-25 Tutorial visitado 1 veces Descargar en PDF

Phonegap/Cordova y las Notificaciones Push.

0. Índice de contenidos.

1. Introducción

Las Notificaciones Push son un mecanismo del que disponen las aplicaciones móviles para poder recibir avisos emitidos por un sistema en cualquier momento siempre que el dispositivo no esté apagado. Las herramientas de trabajo colaborativo, de mensajería, los sistemas de monitorización, redes sociales... son algunos ejemplos de aplicaciones que hacen uso de esta característica.

Las Notificaciones Push son una **funcionalidad nativa** del dispositivo, por lo que si queremos hacer uso de ella deberemos desarrollar una aplicación haciendo uso de una tecnología nativa (Android, IOS...) o cross-platform (Phonegap, Titanium...).

En este tutorial intentaremos explicar mediante un ejemplo cómo funcionan las Notificaciones Push en una aplicación móvil híbrida desarrollada con Apache Cordova para dispositivos Android.

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
- NetBeans IDE 8.0
- Genymotion 2.2.2
- Apache Cordova 3.4.1
- Ionic Framework 1.0.0-beta.9
- Google Cloud Messaging REST Service desplegado en Apache Tomcat 7
- Dispositivos:
  - Samsung Galaxy R (I9103): Android 4.0.4
  - Samsung Galaxy Core: Android 4.1.2
  - Samsung Galaxy Tab: Android 4.0.4
  - Samsung Galaxy S3 Mini: Android 4.1.2

3. ¿Qué vamos a hacer?.

Haremos una pequeña **aplicación móvil híbrida** para Android con ayuda de **Apache Cordova** (si alguien no se ha enterado de nada de lo anterior lo mejor es que pare de leer y le eche un ojo a este tutorial de mi compi Rubén). Nuestra aplicación será capaz de recibir Notificaciones Push desde Google Cloud Messaging. Los requisitos serán los siguientes:

- Las notificaciones podrán ser enviadas a cualquier dispositivo que tenga instalada la aplicación desde cualquier punto de nuestro sistema.
- El usuario de la aplicación debe poder recibir notificaciones en los siguientes casos:
  - El usuario está utilizando la aplicación.
  - La aplicación está corriendo en un segundo plano.
  - La aplicación no se arrancó.
  - El dispositivo está en reposo (pantalla apagada).

Para poder conseguir todo esto, además de hacer uso de Apache Cordova, necesitaremos hacer uso del **plugin de Phonegap para Notificaciones Push**.

4. El plugin de Phonegap para las Notificaciones Push.

Existen diversos plugins de Phonegap/Cordova para Notificaciones Push. Nosotros hemos elegido este por su popularidad y facilidad de uso.

El plugin no da únicamente soporte a la recepción de notificaciones en **Android**, sino también a **IOS**, **Windows Phone 8** y **Amazon Fire OS**. Cada uno recibirá mensajes de sus respectivos servicios en la nube (cada fabricante tiene uno).

El plugin se puede instalar, para cada una de las plataformas, tanto de manera automática como manual. Nosotros por sencillez elegiremos la manera automática (lo veremos en el siguiente punto) que realmente lo que hace es añadir los ficheros nativos al proyecto (.java ya que estamos en Android), definir los permisos necesarios, actividades y servicios. Requisitos necesarios para recibir Notificaciones Push desde Google Cloud Messaging tal y como vimos **en el anterior tutorial**.

5. Creando nuestra aplicación.

Ya vale de teoría, vamos a meternos en el código :-)

5.1 Configuración inicial.

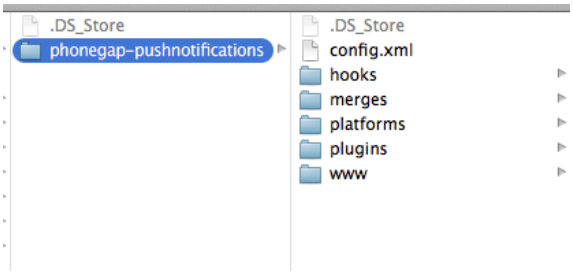
Lo primero que haremos será crear el esqueleto de nuestro proyecto. Para ello, en el directorio deseado ejecutamos el siguiente comando:

```
cordova create . "com.autentia.phonegap.pushnotifications" "PhonegapPush"
```

Donde:

- **.** (punto): El directorio donde queremos crear el proyecto, en nuestro caso en el directorio actual.
- **com.autentia.phonegap.pushnotifications**: Identificador de nuestra aplicación (tipo dominio inverso).
- **PhonegapPush**: El nombre de nuestra aplicación.

Nos creará algo como esto.



Añadimos la plataforma destino de nuestra aplicación. En nuestro caso Android.

```
cordova platform add android
```

Añadimos el plugin de Phonegap que vimos en el punto anterior para recibir Notificaciones Push (instalación automática).

```
cordova plugin add https://github.com/phonegap-build/PushPlugin.git
```

Añadimos el plugin de Cordova para conocer el estado de nuestra conexión. Este plugin es opcional pero nos viene muy bien ya que si nuestro dispositivo no tiene salida a Internet no podremos recibir notificaciones desde GCM.

```
cordova plugin add org.apache.cordova.network-information
```

Añadimos el plugin de Cordova para conocer información relativa al dispositivo. Igualmente es opcional. En nuestro caso queremos saber si el sistema operativo es Android.

```
cordova plugin add https://git-wip-us.apache.org/repos/asf/cordova-plugin-device.git
```

Por último, en nuestro fichero /www/index.html nos aseguramos de que tenemos importada la librería de Javascript de Cordova (cordova.js) y la del plugin de Notificaciones Push (PushNotification.js).

```
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="PushNotification.js"></script>
```

Además utilizaremos un poco de jQuery para que nos facilite un poco el trabajo.

```
<script type="text/javascript" src="js/jquery-1.11.0.min.js"></script>
```

5.2 Registrando el dispositivo en Google Cloud Messaging.

desarrollos Android con Google Cloud Messaging.

Últimos Tutoriales del Autor

» Configurando Notificaciones Push para desarrollos Android con Google Cloud Messaging.

» Introducción a WSO2 API Manager

» SOA vs. SOAP y REST

» REST, el principio HATEOAS y Spring HATEOAS

» SOA y los tipos de servicios

Una vez tenemos la configuración inicial, ya podemos empezar con el código de nuestra aplicación. Lo primero que haremos será registrar nuestro dispositivo en el servicio de Google Cloud Messaging. Si el registro se realizó de manera correcta, obtendremos un identificador de registro (registration ID) tal y como se puede ver en los **pasos 1 y 2** del siguiente diagrama.

Para ello crearemos una función **register** que, haciendo uso del la librería javascript que nos proporciona el plugin de Phonegap para **Notificaciones Push**, nos permitirá poder realizar el registro en GCM (líneas 2 y 5 del siguiente fragmento de código). Además, en el proceso de registro, necesitaremos suministrar dos parámetros (líneas 10 y 11):

- **senderID**: Número de proyecto en Google Cloud Console. Número de proyecto del que ya hablamos y vimos cómo obtener **en el anterior tutorial**
- **ecb**: Nombre de la función a la que el plugin de Notificaciones Push invocará cada vez que reciba un evento desde Google Cloud Messaging. Dicho evento puede ser una **confirmación de registro** (registrationID desde GCM), una **notificación** o un **error**. OJO!!! No puede ser una función de callback, tiene que ser el nombre de la función.

```
1  function register() {
2      var pushNotification = window.plugins.pushNotification;
3
4      if (isAndroidDevice()) {
5          pushNotification.register(function(result) {
6              console.log('Status: ' + result);
7          }, function(result) {
8              alert('Error handler ' + result);
9          }, {
10             "senderID": "000000000000", /* Google developers project number */
11             "ecb" : "onNotificationGCM" /* Function name to handle notifications */
12         });
13     } else {
14         alert('Your device platform is not Android!!!');
15     }
16 }
```

Lo siguiente que haremos será implementar la función **onNotificationGCM** que es el nombre que dimos en el fragmento anterior a la función que manejará las notificaciones que nos lleguen desde GCM.

```
1  function onNotificationGCM(e) {
2      switch (e.event) {
3          case 'registered':
4              if (e.regid.length > 0) {
5                  var registrationId = e.regid; //GCM Registration ID
6                  registerOn3rdPartyServer(registrationId);
7              }
8              break;
9
10         case 'message':
11             // handle notification message
12             break;
13
14         case 'error':
15             // handle error
16             break;
17
18         default:
19             // handle default
20             break;
21     }
22 }
```

Podemos apreciar que cuando el valor de la propiedad **event** del evento que recibimos es **registered** significa que nos hemos registrado en GCM y que éste nos devuelve un identificador de registro (registration ID). Hasta aquí sería lo relativo al los pasos 1 y 2 del anterior diagrama. Con dicho identificador nos registraremos en nuestro servicio (paso 3 del diagrama). Lo vemos en el punto siguiente.

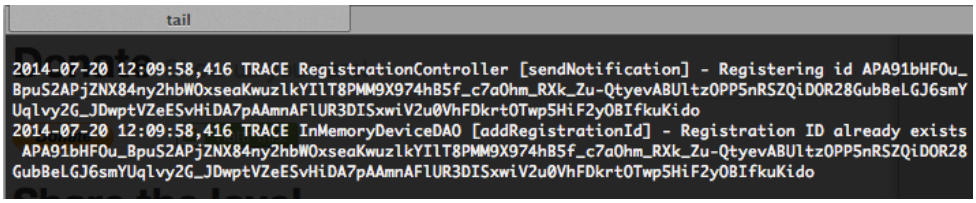
5.3 Registrando el dispositivo en nuestro servidor.

Para que esté totalmente completado el proceso de registro del anterior diagrama, necesitaremos registrar nuestro

dispositivo en nuestro servidor propio. Tan sencillo como enviar una petición HTTP por POST con nuestro identificador de registro (registrationId) que acabamos de recibir de GCM. Él ya se encarga de todo :).

```
function registerOn3rdPartyServer(registrationId) {
    $.ajax({
        type: "POST",
        url: "http://3rd_PARTY_SERVER_HOST:PORT/gcm-rest/api/registrations", /* Your gcm-r
        data: {
            "registrationId": registrationId
        },
        headers : {
            "Content-Type" : "application/x-www-form-urlencoded"
        },
        success: function() {
            statusElement.html('READY FOR NOTIFICATIONS');
        },
        error: function(e) {
            alert("Unable to register " + JSON.stringify(e));
        }
    });
}
```

En el **log del servidor** podremos ver el identificador del registro.



Pues ya estamos registrados, a partir de este punto ya podríamos recibir notificaciones...

5.4 Enviando una notificación.

Ya estamos listos para enviar una notificacion, del mismo modo que hicimos en el punto anterior, aprovecharemos nuestro servicio para realizar dicha tarea.

Muy sencillo, desde cualquier punto de nuestro sistema desde el que tengamos acceso al servidor enviamos una petición como la que se ve a continuación:

```
Url: http://192.168.1.33:8080/gcm-rest/api/notifications
Content-Type: application/json
Method: POST
Request Raw Body:
{
    "badge":1,
    "title":"Notification title",
    "message":"A message",
    "registrationIdsToSend":[
    "APA91bE9f_SCHrrUvTlkibvAyfpk3Ai9YoEIAPhn50tVkryBLolM0RHdbh53tC27VdRcMTWwyervn4zL4Si
    ]
}
```

Donde:

- **badge**: es un número que aparece en la parte derecha de nuestra notificación (lo veremos más abajo). Intenta simular 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**: array con identificadores de registro a los que enviar la notificación. Dichos identificadores deben antes haber sido registrados en nuestro servicio tal y como vimos en el punto anterior.

5.5 Manejando la notificación.

Y lo último que nos queda es manejar la notificación. Para ello, en la función **onNotificationGCM** que vimos en punto 5.2 y que era la responsable de manejar las notificaciones que nos llegasen desde GCM, añadimos la lógica necesaria para procesar el mensaje propio de la notificación.

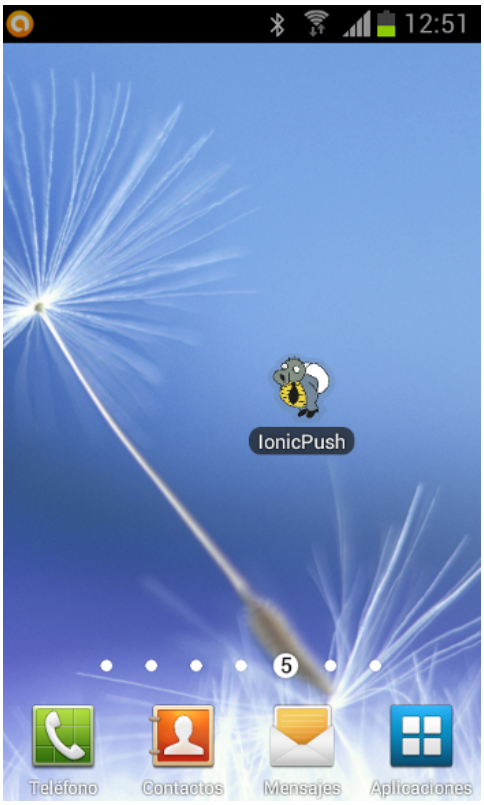
```
function onNotificationGCM(e) {
    switch (e.event) {
        case 'registered':
            // handle registration
            break;

        case 'message':
            if (e.foreground) {
                alert('FOREGROUND MSG:' + JSON.stringify(e));
            } else if (e.coldstart) {
                alert('COLDSTART MSG:' + JSON.stringify(e));
            } else {
                alert('BACKGROUND:' + JSON.stringify(e));
            }
            break;

        case 'error':
            // handle error
            break;

        default:
            // handle default
            break;
    }
}
```

Dentro de la información de la notificación podremos saber el estado de la aplicación al recibir el mensaje (por si queremos manejarlo de diferentes formas):

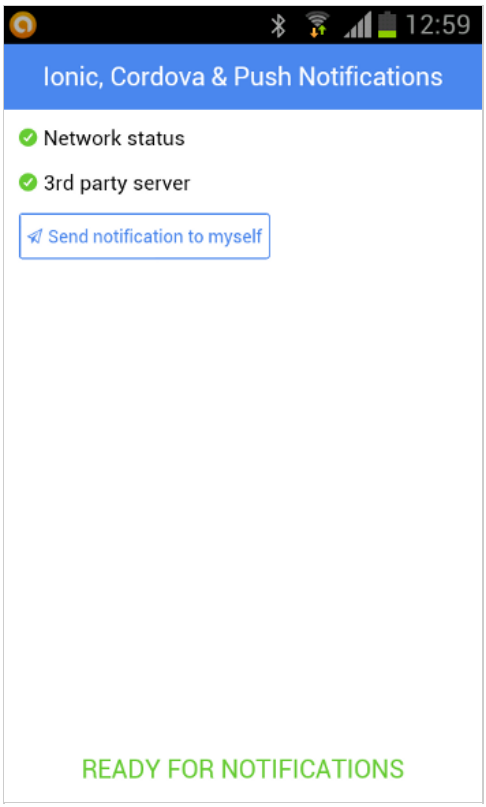
- **foreground**: cuando el usuario está utilizando la aplicaicón.
- **coldstart**: cuando la aplicación estaba en un segundo plano.
- **todo lo demás**: la aplicación arranca por primera vez.

6. El resultado final.

Pues utilizando como base todo esto que hemos explicado, un poco de **AngularJS** y una pizca de **Ionic** (un excelente framework HTML5 para desarrollar aplicaciones móviles híbridas) ya **tendríamos nuestra aplicación lista** para recibir Notificaciones Push desde Google Cloud Messaging :D.

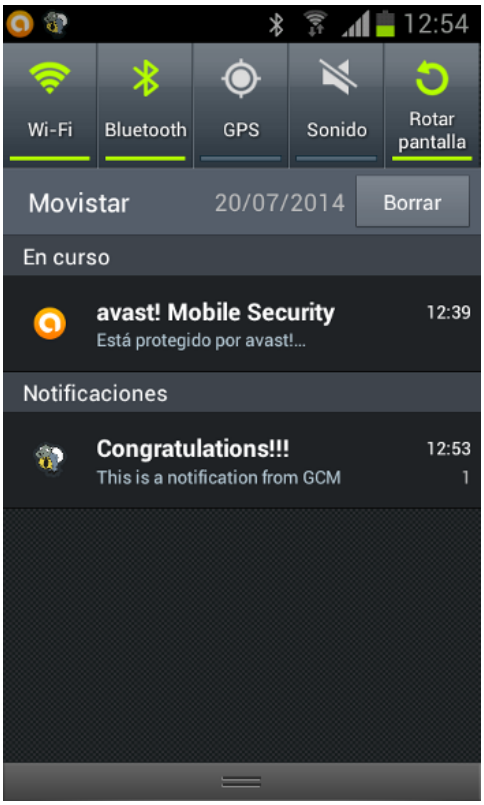


La aplicación arranca y comprueba que tenemos acceso a Internet, se registra en Google Cloud Messaging y en nuestro servidor.

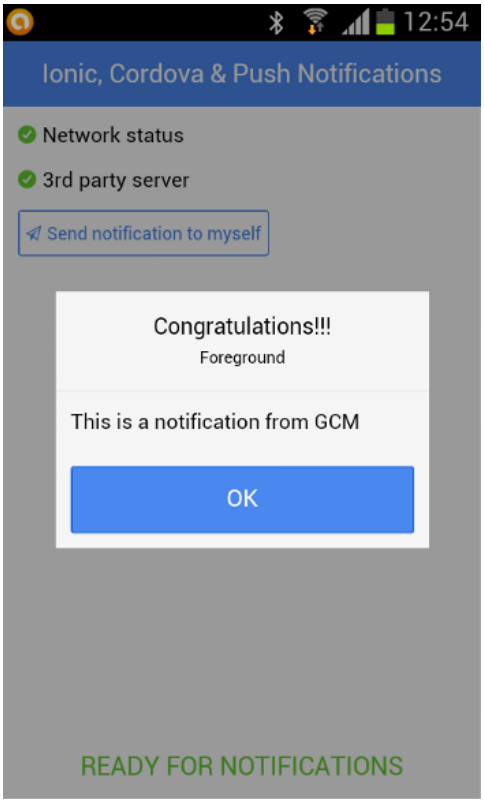


Si **cerramos o minimizamos nuestra aplicación, incluso si ni siquiera la hubiésemos arrancado o si el móvil estuviese en reposo**, al recibir una notificación nos aparecerá el icono de la aplicación indicando que tenemos una nueva notificación.

Desplegando el menú de opciones vemos más en detalle nuestra notificación. El 1 que se ve en la parte derecha es el parámetro **badge** que enviamos en el punto anterior.



Si pulsamos sobre la propia notificación, se abrirá la aplicación y nos saltará un mensaje con el texto y el título. Si estuviésemos utilizando la aplicación (foreground) y nos llegase otra, nos saltaría de nuevo el mensaje pero no nos aparecerá el icono de la parte superior.



AQUÍ OS DEJO TODO EL CÓDIGO FUENTE DE LA APLICACIÓN

7. Referencias.

- Código fuente del ejemplo.
- Configurando Notificaciones Push para desarrollos Android con Google Cloud Messaging
- Google Cloud Messaging REST Service
- Habilitar Genymotion para Notificaciones Push

8. Conclusiones.

En este tutorial hemos visto cómo utilizar el plugin de Notificaciones Push que nos proporciona Phonegap para crear una aplicación híbrida (corre en un webview) que, interactuando de manera nativa con el dispositivo, pueda ser capaz de recibir notificaciones en cualquier momento desde Google Cloud Messaging.

El **desarrollo híbrido o cross-platform** en términos de movilidad puede tener algunas desventajas con respecto al desarrollo nativo pero también tiene muchas otras **ventajas**. Desde luego, la recepción de notificaciones remotas ya no será un problema si utilizamos un framework como Apache Cordova (o incluso otros como Titanium, pero ese es otro tema...).

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

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