



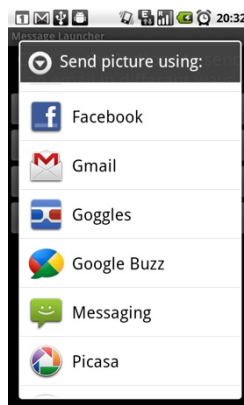
CAPÍTULO 8

Redes sociales: Facebook y Twitter

Por JORDI BATALLER MASCARELL

Es difícil hoy en día no estar vinculado a una red social donde mantener el contacto con amigos y conocidos, compartir inquietudes, aficiones, deportes, juegos y opiniones. Con sus matices, Facebook y Twitter nos ofrecen darnos a conocer y conocer a otros. Por ello, es casi obligatorio que también los programas que desarrollemos puedan usar estas redes sociales para obtener o mostrar información de quien los está utilizando.

Entrando en el terreno del desarrollo en Android, antes de programar hay que pensar cuál es la forma más sencilla de resolver el problema que tenemos entre manos, evaluando varias alternativas. En el caso de querer publicar contenido en una red social, la primera pregunta que debemos hacernos es si lanzar una intención implícita “SEND” es suficiente para nuestros propósitos, de forma que otra aplicación distinta de la nuestra se encargue de ello:



Si en cambio, lo que queremos es:

- integrar el acceso y uso de una red social directamente en nuestra aplicación
- aprovechar el login de la red social para validar los usuarios de nuestra aplicación, ahorrándonos el trabajo de mantener usuarios y contraseñas (con las correspondientes altas, bajas y modificaciones) y evitando que programemos nuestro proceso propio de login/logout.

sigamos leyendo.

**Objetivos:**

- Conseguir una cuenta de desarrollador en Facebook y Twitter.
- Aprender a utilizar la «consola» de gestión de aplicaciones en estas dos redes sociales.
- Dar de alta la aplicación que queremos desarrollar.
- Descargar y configurar las bibliotecas que nos servirán para interactuar con las redes sociales.
- Configurar y programar una aplicación integrada en Facebook y Twitter.

En esta sección aprenderemos dar de alta, configurar y desarrollar una aplicación que pueda interactuar *directamente* con Facebook. La parte de configuración es relativamente larga y debe ser seguida detalladamente. La propia aplicación en Facebook también tiene sus peculiaridades, como que las llamadas son asíncronas, cosa que afecta a la estructura de la aplicación.

8.1. Android y Facebook

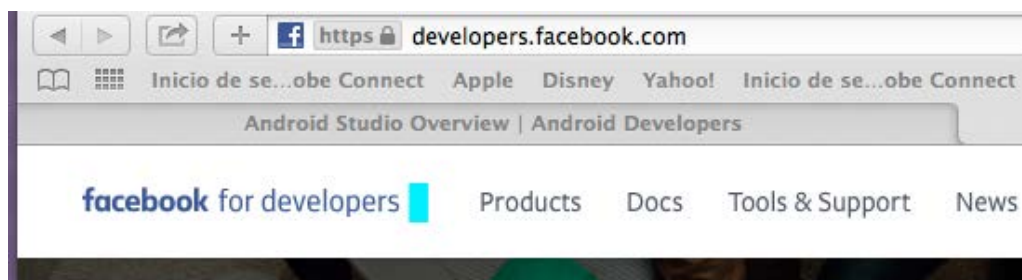
8.1.1. Preliminares

En primer lugar vamos a ver cómo dar de alta nuestra aplicación, configurarla y descargar el kit de desarrollo propio de Facebook.

8.1.1.1. Darse de alta en Facebook como desarrollador

Para escribir aplicaciones para Facebook hay que tener un usuario en la red social, pero además hay que darse de alta como desarrollador. Accedemos a

<http://developers.facebook.com/>



y vamos a Aplicaciones (Apps), donde empezará el proceso de registro.



Android: Programación Avanzada

En él se nos pide un número de teléfono al que se envía una clave, que utilizaremos para completar el proceso. Más tarde volveremos a “Apps” para dar de alta nuestra aplicación.



Ejercicio: *Date de alta en Facebook developers y explora el sitio.*

8.1.1.2. Aplicación oficial de Facebook para Android

Es necesario instalar la aplicación oficial de Facebook en el teléfono o en el emulador que vayamos a utilizar.



En el caso de los teléfonos, muchos la tienen pre-instalada o pueden instalarla fácilmente desde “Google Play”. Si vamos a utilizar un emulador, deberemos conseguir el .apk: podremos hacerlo al dar de alta nuestra aplicación en la consola de desarrollo de Facebook (ver más abajo) o en <https://developers.facebook.com/docs/android/downloads>. Dicho .apk se puede instalar escribiendo esta orden en la consola: (estando el emulador en ejecución):

```
adb install Facebook-35.0.0.48.273.apk
```

8.1.1.3. SDK de Facebook para Android

Facebook proporciona un SDK (kit para desarrollo de software) para escribir aplicaciones que interactúen con su plataforma. Anteriormente debíamos descargar este SDK manualmente para utilizarlo en nuestros programas. Sin embargo, actualmente no es necesario hacerlo



Android: Programación Avanzada

porque cambiando la configuración del fichero “build.gradle” de nuestro proyecto, éste será capaz de realizar la descarga automáticamente. Si necesitáramos descargarlo manualmente, podemos hacerlo en el siguiente enlace: <https://developers.facebook.com/docs/android/>

SDK de Facebook para Android

Te ayuda a crear aplicaciones sociales atractivas y aumentar el número de descargas.

[Descargar SDK](#)

Incluye los paquetes de Account Kit, Audience Network y Facebook. Requiere la API 15 de Android.

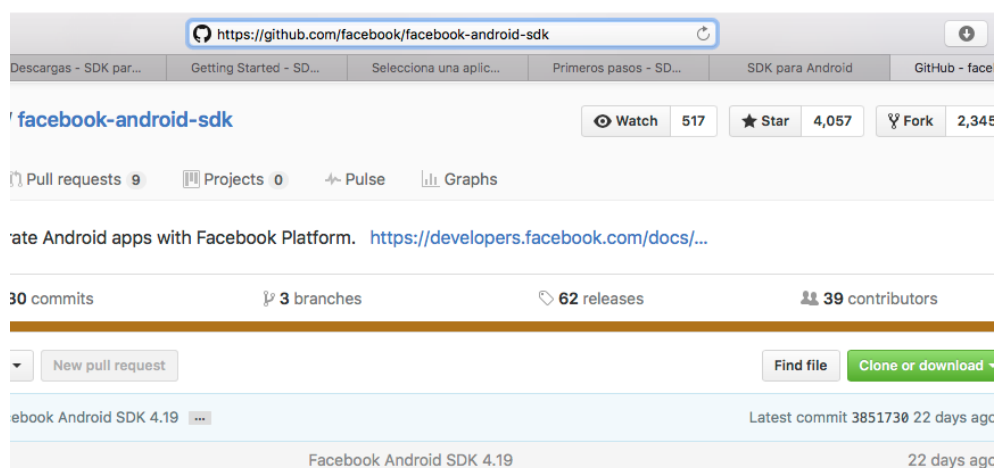
v4.19.0. Consulta el [registro de cambios](#) o la [guía de actualización](#).

[Primeros pasos](#)

Guía básica para Android

8.1.1.4. Ejemplos básicos de Facebook SDK

Podemos encontrar el código fuente de una serie de ejemplos en el siguiente repositorio de github: <https://github.com/facebook/facebook-android-sdk>



Podemos configurar Android Studio para que acceda a los ejemplos, o descargarlos en un zip mediante el botón “Clone or download”.

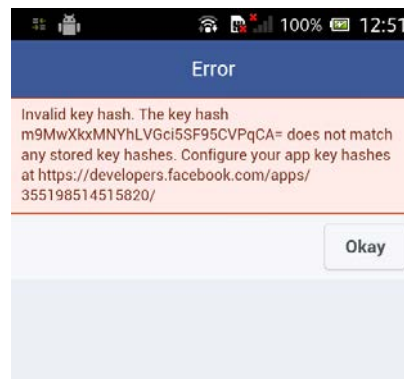


Ejercicio: Descarga de github el código de las aplicaciones de ejemplo de Facebook.

Conviene probar el ejemplo “HelloFacebookSample” para ver que todo va correctamente antes de realizar nuestros propios programas. Para ello, sólo hay que abrir ese ejemplo donde lo hayamos descargado. Para que el ejemplo funcione correctamente hay que registrar la clave pública (“keyhash”) con la que se firman nuestras aplicaciones (ficheros .apk) en “Facebook Developers”. Si no realizamos este paso, al utilizar, por ejemplo, la aplicación “HelloFacebookSample”, veremos en algún momento el siguiente error:



Android: Programación Avanzada



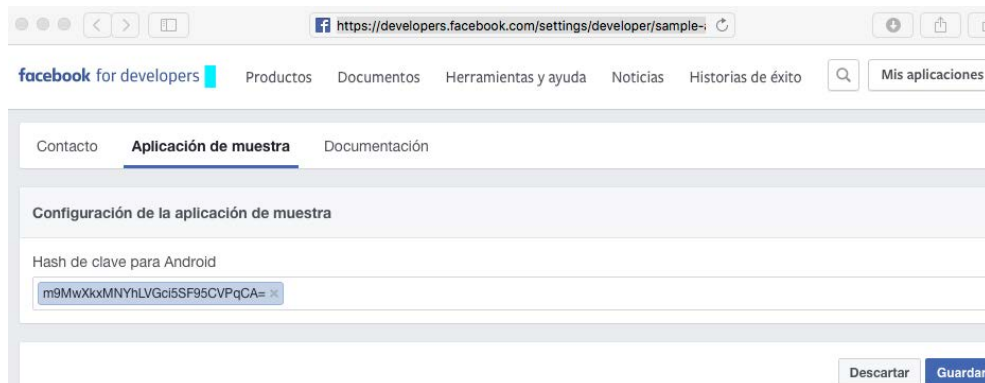
O en forma de “toast”:

(#404) Key hash
lpgCWdGg47PEaDQy_aO-54rt_TU does not
match any stored key hashes.

Por tanto, para poder ejecutar las aplicaciones de ejemplo consultaremos la clave pública con que se firman nuestras aplicaciones (.apk) y que justamente aparece mensaje del error. La clave está en “~/android/debug.keystore” y se averigua utilizando los comandos *keytool* y *openssl*, (ver siguiente sección). Cuando sepamos esa clave, iremos a

<https://developers.facebook.com/settings/developer/sample-app>

Elegimos la pestaña “Sample App” (‘‘aplicación de muestra’’) y en “Identificador de clave Android (Android Key Hash)” copiamos nuestra clave pública, dando a “Guardar cambios (Save Changes)”.



En general, necesitaremos recompilar las aplicaciones de ejemplo de Facebook para que integren la clave a partir de “debug.keystore”, si la hemos cambiado.

Por fin, podremos probar el ejemplo “HelloFacebookSample”:



 Continue with Facebook



Con esta aplicación podremos hacer “login” en Facebook y enviar mensajes y fotos a nuestro muro. Cuando la aplicación pida permisos aparecerá una pantalla para que nos identifiquemos, solicitando el nombre de usuario y password de Facebook. Esto no es necesario si tenemos la aplicación oficial de Facebook abierta con nuestro usuario, o ya hemos dado los permisos con anterioridad.



Ejercicio: Prueba la aplicación HelloFacebookSample.

Recuerda poner la clave keyhash de tus .apk en el apartado “Sample App/Aplicación de Muestra” de Facebook developers tal como se ha descrito antes. En ocasiones, según la versión del navegador la clave no se queda guardada. Si te ocurre este bug, prueba con varios navegadores y/o sistemas operativos.

8.1.1.5. Configurando nuestra nueva aplicación (en Facebook Developers)

Después comprobar que la aplicación de ejemplo funciona, vamos a tratar de escribir una aplicación nuestra que haga algo similar, tomando código de los ejemplos.

El primer paso, antes de utilizar Android-Studio, es dar de alta en nuestro gestor de “Apps” de Facebook developers los datos de la aplicación que queremos desarrollar, aunque esta no esté aún escrita.

Para ello, vamos a developers.facebook.com/apps, donde pulsaremos en el Botón “Añadir una nueva aplicación” (o también en “My Apps → Add a New App”/Mis Aplicaciones→ Añadir una nueva aplicación”).

Se nos preguntará datos básicos como el nombre de la aplicación, un correo electrónico, el tipo de aplicación, etc.



Android: Programación Avanzada

Facebook para desarrolladores | Primeros pasos - SDK para Android | Todas las aplicaciones - Facebook para desarrolladores

Productos | Documentos | Herramientas y ayuda | Noticias | Historias de éxito | Buscar | Mis aplicaciones

+ Añadir una nueva aplicación

Crear un nuevo identificador de la aplicación

Empieza a integrar Facebook en tu aplicación o sitio web

Nombre para mostrar
HelloFacebookSample2017

Correo electrónico de contacto
Se usa para comunicaciones importantes sobre tu aplicación

Categoría
Educación ▼

Al continuar, aceptas las Normas de la plataforma de Facebook

Cancelar | Crear identificador de la aplicación

Terminado este paso, estaremos en el “zona de control” de nuestra nueva aplicación.



Si pulsamos en la pestaña “Panel” de la derecha veremos



Android: Programación Avanzada



Aquí debemos pulsar en el botón “Elegir plataforma” para elegir “Android” en la siguiente pantalla



Tras elegir “Android” aparece una guía que nos dice los pasos de configuración que hay que realizar para terminar de configurar nuestra aplicación.

Como es lo que vamos a explicar ahora, pulsamos directamente “Skip Quick Start” (esquina superior derecha). Al hacerlo, seguimos estando aquí:



Fijémonos en el “Identificador de la aplicación / App ID” que nos ha asignado Facebook y en la “clave secreta” porque luego los vamos a necesitar. En “Configuración / Settings”, también podemos encontrar esos datos. Allí se puede poner o cambiar el correo electrónico de contacto así como la imagen o icono de nuestra aplicación.



Android: Programación Avanzada

Panel

Configuración

Información básica

Opciones avanzadas

Roles

Alertas

Revisión de la aplicación

PRODUCTOS

+ Añadir producto

IDENTIFICADOR DE LA APLICACIÓN: 1713136355643667

Herramientas y ayuda

Documentos

Identificador de la aplicación: 1713136355643667

Clave secreta de la aplicación: 7dd263b63d07a17e6ccc8234284cad88

Nombre para mostrar: HelloCarLibroSample2017

Espacio de nombres:

Dominios de aplicaciones:

Correo electrónico de contacto: Se utiliza para comunicaciones importantes sobre

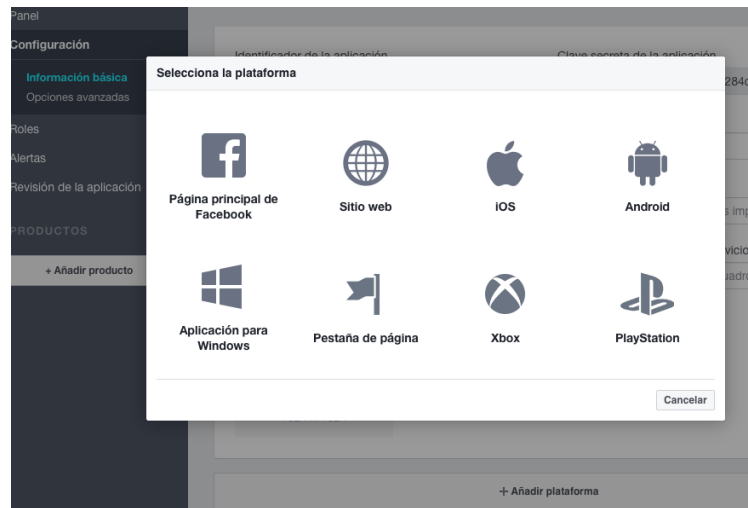
URL de la política de privacidad: Política de privacidad del cuadro de diálogo de inicio de

URL de las Condiciones del servicio: Condiciones del servicio del cuadro de diálogo

Icono de la aplicación (1024 x 1024):

Categoría: Educación

Aquí debemos de nuevo elegir la plataforma (o plataformas, ya que podemos tener la misma aplicación para varios sistemas) en que vamos a desarrollar. Lo hacemos pulsando el botón “+ Elegir plataforma” que nos da para elegir:



Tras haber elegido “Android”, aparece una nueva zona de datos en la pantalla que es donde deberemos escribir el nombre de la clase principal de nuestra aplicación y especialmente la clave (keyhash) con que estará firmado su correspondiente .apk.

Android

Inicio rápido

Nombre del paquete de Google Play: org.jordi

Nombre de la clase: HelloCarLibroSample

Hashes de clave: m8MwX0uMNYhLVGc5SP85CVpQCA=

URL de Tienda Apps de Amazon (opcional):

Generar APK

Para poder consignar en la anterior pantalla la parte pública de la clave con que se firman nuestros ficheros .apk, deberemos averiguarla. Las claves están guardadas en %HOMEPATH%\.android/debug.keystore y se obtiene con los comandos “keytool” y “openssl” :



Android: Programación Avanzada

```
keytool -exportcert -alias androiddebugkey -keystore  
%HOMEPATH%\android\debug.keystore | openssl sha1 -binary | openssl base64  
Escriba la contraseña del almacén de claves: android
```

Véase que la contraseña que hemos de dar es **android**. El anterior comando nos devolverá una clave pública como la siguiente (acabada en con el símbolo =)

```
IpgCwDgG47PEaDQy/a0+54rt/TU=
```

Nota: En Unix, el comando es el mismo cambiando %HOMEPATH% por \$HOME y las barras \ por /. La utilidad “keytool” está instalada con el JDK de Java. La utilidad “openssl” puede descargarse gratuitamente desde internet (buscar en google). Ambas hay que añadirlas a la variable de entorno PATH.

El keyhash se puede averiguar también a partir de un .apk. En Mac-OSX el comando es

```
keytool -list -printcert -jarfile miaplicacion.apk | grep "SHA1: " | cut -d " " -f 3  
| xxd -r -p | openssl base64
```

En linux/unix/MacOS es:

```
keytool -list -printcert -jarfile [path_to_your_apk] |  
grep -Po "(?<=SHA1:).*" | xxd -r -p | openssl base64
```

En windows se puede utilizar estos comandos, habiendo instalado antes las herramientas grep, cut y xxd. Finalmente, también es posible extraer el keyhash por programa mediante el siguiente código:

```
PackageInfo info;  
try {  
    info = getPackageManager().getPackageInfo("com.you.name", // nombre paquete  
        PackageManager.GET_SIGNATURES);  
    for (Signature signature : info.signatures) {  
        MessageDigest md;  
        md = MessageDigest.getInstance("SHA");  
        md.update(signature.toByteArray());  
        String something = new String(Base64.encode(md.digest(), 0));  
        // Log.d("KeyHash:", Base64.encodeToString(md.digest(), Base64.DEFAULT));  
  
        Log.e("hash key", something);  
    }  
} catch (NameNotFoundException e1) {  
    Log.e("name not found", e1.toString());  
} catch (NoSuchAlgorithmException e) {  
    Log.e("no such an algorithm", e.toString());  
} catch (Exception e) {  
    Log.e("exception", e.toString());  
}
```

Este paso suele dar problemas y más tarde la aplicación no puede publicar en Facebook o cualquier otra actividad relacionada. Recomendamos fijarse que no aparece ningún mensaje de aviso (“warning”) o error en la consola al extraer la clave. También es conveniente que extraigamos más de una vez la clave y comprobemos que es la misma siempre. Si uno de los comandos falla, al final lo que obtendremos será un mensaje de aviso o error codificado como clave hash, en lugar de la auténtica clave guardada.

Cuando ya estemos seguros de tener la clave correcta, en Facebook developers, vamos a la sección de Android de la configuración y copiamos la nuestra clave en “Identificadores de clave (Key hashes)” y guardamos los cambios.



Android: Programación Avanzada



Los campos “nombre del paquete” (“Google Play Package Name”) y “Clase” (“Class Name”) los rellenaremos cuando generemos el proyecto de Android-Studio. Es importante recordar que la clave la tenemos guardada localmente en nuestro ordenador (en “debug.keystore”), por lo que si cambiamos de ordenador, o generamos una .apk para distribuir (“release”: que utiliza las claves de “release.keystore”) deberemos repetir este paso, averiguando la clave correspondiente y añadiéndola como acabamos de describir.



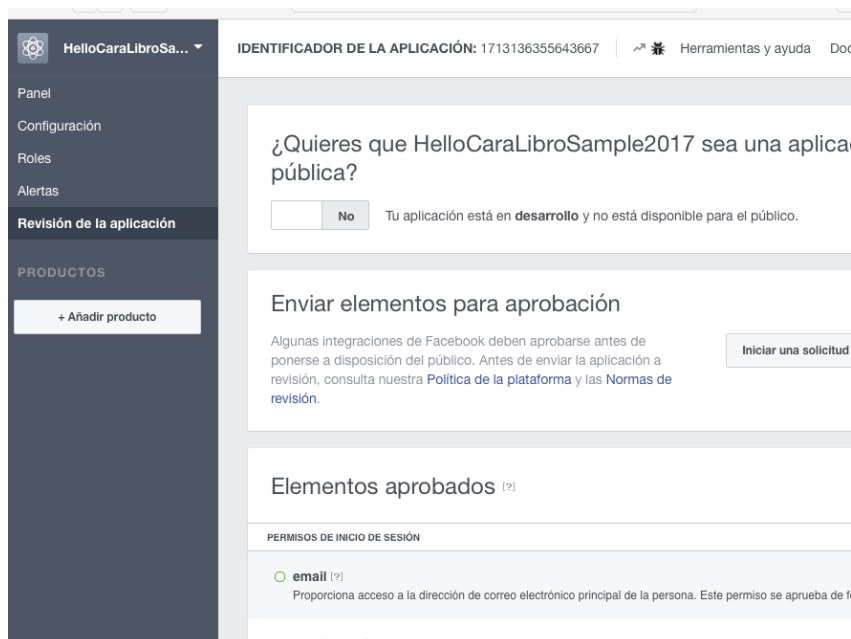
Ejercicio: Aprende a extraer el keyhash de tu debug.keystore de alguna de las formas antes explicadas.



Ejercicio: Da de alta una nueva aplicación tuya en Facebook developers siguiendo las anteriores instrucciones.

8.1.1.6. Algunas consideraciones “Sandbox mode”, “approved items”

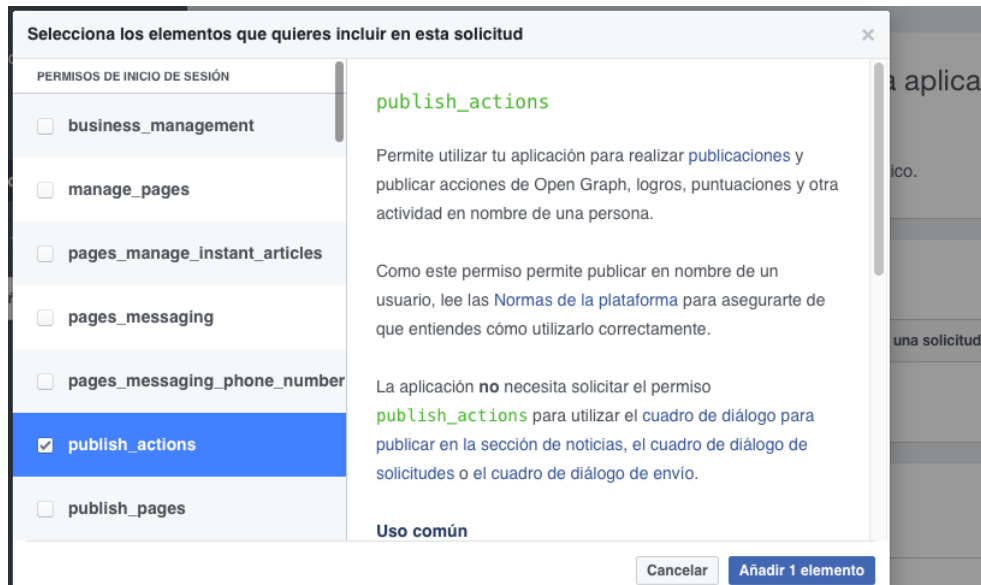
Si se quiere que la aplicación sea utilizada por cualquier usuario además del desarrollador, hay que desactivar “sandbox mode” haciendo que sea una aplicación pública. Esto puede hacerse en “Revisión de la aplicación / Status & Review” marcando “Sí” en “¿Quieres que sea una aplicación pública”.





Las aplicaciones, por defecto, tienen derecho a solicitar al usuario permiso para “email”, “public_profile” (información básica sobre el usuario) y “user_friends” (lista de amigos del usuario). El usuario concederá estos permisos al hacer login en Facebook desde nuestra aplicación.

Desde 2015, si nuestra aplicación desea realizar acciones u obtener información diferente de las anteriores, (p.ej. pedir permiso para publicar: “publish_actions”), deberemos solicitar a Facebook que revise nuestra aplicación y la valide.

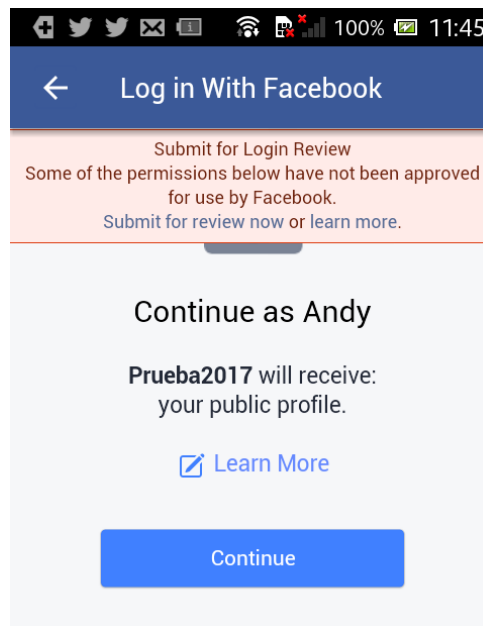


Es decir: **deberemos informar y solicitar a Facebook autorización para poder pedir permiso al usuario sobre determinadas capacidades.** En este mismo apartado, podremos pulsar “Iniciar una solicitud / Start a submission” para conseguirlo. No es un proceso inmediato: hay que proporcionar información a Facebook y esperar entre 4 y 7 días laborables a que respondan.

En resumen, por defecto (sin que Facebook valide nuestra aplicación) sólo podremos publicar desde nuestra aplicación si el usuario Facebook que la utiliza es el mismo usuario que como desarrollador ha dado de alta la aplicación. Insistimos: para que un usuario distinto del desarrollador pueda publicar mensajes, el desarrollador tendrá que haber desactivado el “sandbox mode” y que su aplicación haya sido revisada y aprobada para el público general. Mientras estemos en esta situación podremos ver avisos como el siguiente:



Android: Programación Avanzada



Entendamos que toda esta protección es para evitar comportamientos incorrectos de una aplicación no validada (p. ej. que publique en nombre de un usuario, mensajes que éste no ha escrito).

API Graph

La biblioteca del SDK de Facebook tiene un grupo de funciones denominadas *API Graph* que permiten interactuar *directamente* con Facebook. Se denomina así porque usa la metáfora de que una red social es un grafo: nodos que son personas o recurso y aristas que conectan las personas y los recursos entre ellas (se puede encontrar información sobre este API en <https://developers.facebook.com/docs/graph-api/overview/>). Esta api es de tipo REST: utiliza verbos de HTTP como GET, POST, DELETE para ejecutar acciones sobre los recursos e información almacenada en Facebook. Por ejemplo, para obtener información sobre el usuario actual ("me" en inglés) la petición sería:

```
GET /v2.8/me HTTP/1.1
```

Sin embargo, trabajar a este nivel de detalle es extremadamente complicado porque la petición HTTP hay que completarla con muchos más campos que incluyen información (tokens) de validación. El SDK de Facebook para Android envuelve dichas peticiones dentro de métodos mucho más sencillos de usar. En cualquier caso hay una herramienta de exploración de esta API en este enlace: <https://developers.facebook.com/tools/explorer/>. En la siguiente pantalla podemos ver qué orden REST es la que permite enviar un nuevo mensaje ("status") a nuestro muro.



Android: Programación Avanzada

facebook for developers

Aplicación: [?] Graph API Explorer

Identificador de acceso: EAACEdEose0cBALJCMH3CcZBYqa7sfZAtp5zKR8kQxj9Fy8WlvUOWSqEG\ Obtener ...

POST → /v2.8 → /me/feed?message=hola

Name Value

Add a Field

Obtén más información sobre la sintaxis de la API Graph

```
{
  "id": "100005015110084_710042735839557"
}
```

Al final de la anterior página hay un botón para obtener el código que utilizaríamos para dicha acción según la plataforma de desarrollo. Por ejemplo, para Android veríamos:

```
SDK para Android SDK para iOS SDK para JavaScript SDK para PHP cURL
```

```
GraphRequest request = GraphRequest.newPostRequest(
    accessToken,
    "/me/feed",
    new JSONObject("{\"\":\"\"}"),
    new GraphRequest.Callback() {
        @Override
        public void onCompleted(GraphResponse response) {
            // Insert your code here
        }
    });
request.executeAsync();
```

Learn more about the SDK para Android.

recibido una respuesta en 638 ms

Copy Debug Information Obtener código

Share Dialog

A parte del Graph API que permite interactuar directamente con Facebook, la biblioteca de Facebook tiene otra funcionalidad para interactuar *indirectamente* con Facebook llamada *Share Dialog*. Esta forma no requiere login ni validación de nuestra aplicación porque en realidad utilizan la app oficial de Facebook para Android (determinadas actividades suyas). En esta alternativa, el usuario ya ha hecho login en la propia app de Facebook (no en nuestra aplicación), y cada vez que nuestra aplicación requiera algo, aparecerá la ventana correspondiente de la app de Facebook.

Gestión de las aplicaciones “instaladas” en Facebook



Android: Programación Avanzada

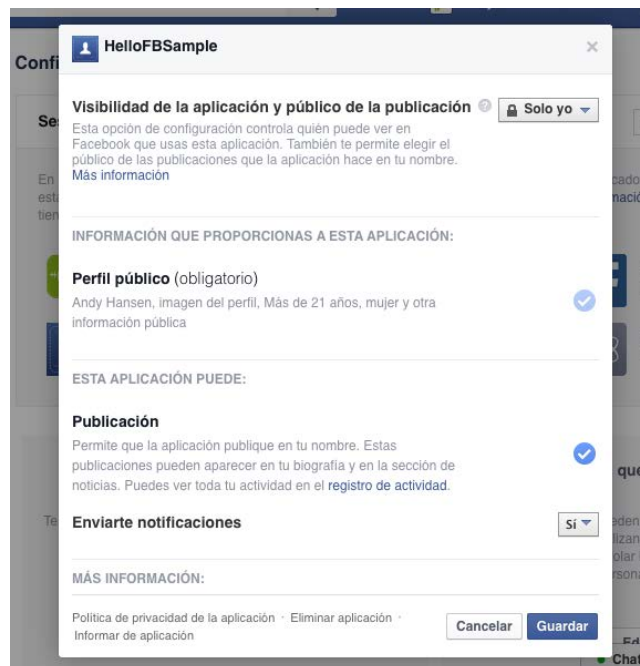
Un usuario corriente de Facebook (*no como desarrollador*) puede ver qué aplicaciones tienen permiso para interactuar en su nombre ante Facebook. Para ver cuáles son, vamos a “Configuración”



y luego pinchamos en la pestaña “aplicaciones”, donde veremos la lista de aplicaciones autorizadas en nuestro nombre. Por ejemplo:



Desde aquí podemos ver qué permisos utiliza cada aplicación relacionada con Facebook que estamos utilizando (en cualquier dispositivo) y revocar o conceder permisos. Por ejemplo:



8.1.2. Nuestro proyecto Android

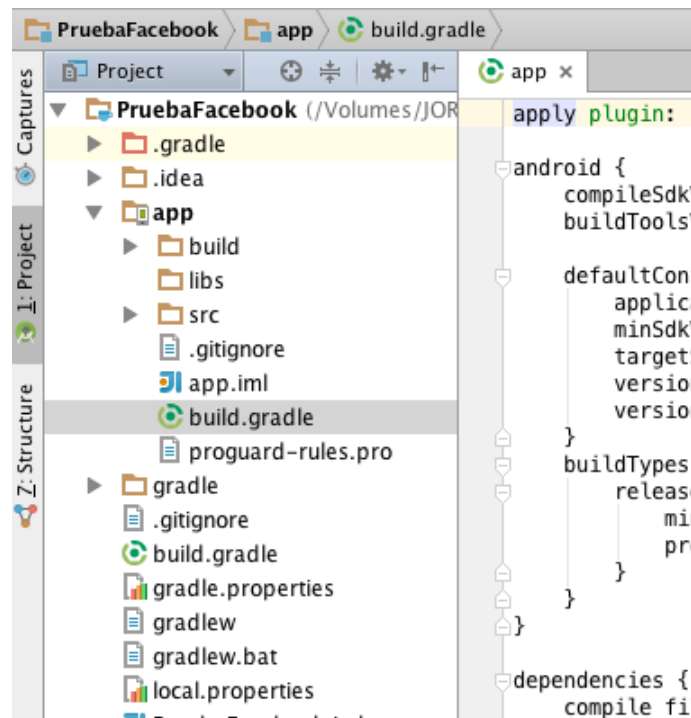
Tras haber realizado la descripción de nuestro programa ante Facebook, a falta de un pequeño detalle, es el momento de crear un proyecto para desarrollar en Android nuestra aplicación.

8.1.2.1. Importando la biblioteca Facebook-Android-SDK en Android Studio

Para utilizar el SDK de Android en un proyecto **nuevo**, tendremos que abrir el fichero build.gradle (sección app)



Android: Programación Avanzada



y añadir dentro de la sección `android { }`:

```
repositories {  
    mavenCentral()  
}
```

y dentro de la sección `dependencies { }`

```
compile 'com.facebook.android:facebook-android-sdk:[4,5)'
```

Esta modificación hará que automáticamente se conecte al repositorio “maven central” de internet para descargar la biblioteca de Facebook.

8.1.2.2. Configurando nuestra aplicación (en Facebook)

Antes ya habíamos declarado en Facebook cual es la clave con que firmamos nuestra aplicación. Ahora que ya hemos empezado nuestra aplicación con Android Studio, podemos copiar el nombre de la clase Java donde la estamos escribiendo. Si por ejemplo ésta se llama

```
org.jordi.pruebafacebook2017sdk1
```

entonces, trasladaremos ese nombre a la sección “Configuración/Setting” de nuestra aplicación:

Android: Programación Avanzada

Prueba2017 IDENTIFICADOR DE LA APLICACIÓN: 321610988240930 Ver Analytics Herramientas y ayu

Panel Configuración Información básica Opciones avanzadas Roles Alertas Revisión de la aplicación PRODUCTOS Inicio de sesión con Facebook + Añadir producto

Icono de la aplicación (1024 x 1024) Categoría Educación

Android Inicio rápido

Nombre del paquete de Google Play org.jordi.pruebafacebook2017sdk1 Nombre de la clase org.jordi.pruebafacebook2017sdk1.MainActivity

Hashes de clave m9MwXioxMNYhLVGci5SF95CVPqCA=

8.1.2.3. Configurando el código de nuestra aplicación

El siguiente paso, consiste en guardar el número identificador con que nuestra aplicación está registrada en Facebook. Podemos ver dónde encontrarlo en la anterior pantalla.

Así pues, en `res/values/strings.xml` de nuestro proyecto añadimos la propiedad `"facebook_app_id"` con el valor correspondiente:



En el fichero `"AndroidManifest.xml"` hay que realizar cambios, relacionados con el acceso a internet que necesita nuestra aplicación, con la identificación de nuestra aplicación Facebook, y con el proceso de login. Reproducimos a continuación el contenido del `"AndroidManifest.xml"` indicando en los comentarios los 3 cambios necesarios:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.jordi.pruebafacebook2017sdk1">

    <!-- 1. añadir estos dos permisos -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
```



```
<!-- 2. añadir por exigencia de Facebook -->
<meta-data android:name="com.facebook.sdk.ApplicationId"
android:value="@string/facebook_app_id" />

<!-- nuestra actividad -->
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>

<!-- 3. añadir actividad para hacer login en Facebook (por su exigencia) -->
<activity android:name="com.facebook.FacebookActivity" />

</application>

</manifest>
```

8.1.3. Aplicación de ejemplo (usando API Graph)

La aplicación que vamos a estudiar como ejemplo simplemente sirve para hacer login en Facebook y publicar mensajes o imágenes en él (utilizando el Graph API), en nombre de un usuario acreditado (que sólo podrá ser, si no pedimos a Facebook que valide nuestra aplicación, el programador):



Android: Programación Avanzada



Para hacer login, podemos utilizar el botón oficial de Facebook y, como ejemplo adicional, unos botones ordinarios. El proceso de login lleva a la aplicación oficial de Facebook de nuestro teléfono informando al usuario sobre los permisos requeridos. Mientras desde allí (aplicación oficial) no hagamos logout, no se pide nuevamente ninguna contraseña en nuestra aplicación para hacer login. Si ya estábamos identificados con la aplicación oficial, hace el login directamente.

Detalles del código

- En primer lugar, para incluir el botón de Facebook hay que incluir en layout de la actividad lo siguiente.

```
<com.facebook.login.widget.LoginButton
    android:id="@+id/login_button"
    android:layout_width="wrap_content"
    android:layout_height="121dp"
    facebook:com_facebook_confirm_logout="false"
    facebook:com_facebook_tooltip_mode="never_display"
/>
```

- Hay que configurar el botón de Facebook según los permisos que necesitamos:

```
loginButtonOficial.setPublishPermissions("publish_actions");
```

- Conviene registrar un callback para saber si el login tiene éxito:

```
LoginManager.getInstance().registerCallback(this.elCallbackManagerDeFacebook,
    new FacebookCallback<LoginResult>() {
        @Override
        public void onSuccess(LoginResult loginResult) { ...
```



Android: Programación Avanzada

- Debemos sobrescribir onActivityResult() y allí informar a Facebook:

```
this.elCallbackManagerDeFacebook.onActivityResult(requestCode, resultCode, data);
```

- Una forma sencilla de obtener el profile (datos básicos) del usuario acreditado:

```
Profile profile = Profile.getCurrentProfile();
```

- Cuando un usuario está acreditado, nuestra aplicación dispone de un "access token" que nos representa ante Facebook, y que es necesario para interactuar con él. Se puede obtener así:

```
AccessToken accessToken = AccessToken.getCurrentAccessToken();
```

- Si no utilizamos el botón de login de Facebook, podemos también hacer login o logout de esta forma:

```
LoginManager.getInstance().loginWithPublishPermissions(this,  
    Arrays.asList("publish_actions"));  
LoginManager.getInstance().logout();
```

- Enviar mensaje al muro de Facebook, mediante API Graph:

```
Bundle params = new Bundle();  
params.putString("message", textoQueEnviar);  
  
GraphRequest request = new GraphRequest(  
    AccessToken.getCurrentAccessToken(),  
    "/me/feed",  
    params,  
    HttpMethod.POST,  
    new GraphRequest.Callback() {  
        public void onCompleted(GraphResponse response) {  
            Toast.makeText(THIS, "Publicación realizada: " + textoQueEnviar,  
                Toast.LENGTH_LONG).show();  
        }  
    }  
);  
request.executeAsync();
```

- Enviar imagen al muro de Facebook, mediante API Graph:

```
Bundle params = new Bundle();  
params.putByteArray("source", byteArray); // bytes de la imagen  
params.putString("caption", comentario); // comentario  
  
// si se quisiera publicar una imagen de internet: params.putString("url",  
// "{image-url}");  
  
GraphRequest request = new GraphRequest(  
    AccessToken.getCurrentAccessToken(),  
    "/me/photos",  
    params,  
    HttpMethod.POST,  
    new GraphRequest.Callback() {  
        public void onCompleted(GraphResponse response) {  
            Toast.makeText(THIS, "" + byteArray.length + " Foto enviada: " +  
                response.toString(), Toast.LENGTH_LONG).show();  
        }  
    }  
);  
request.executeAsync();
```



Android: Programación Avanzada

Hay que hacer notar que la Graph API de Facebook es una interfaz tipo REST que “por debajo” envía peticiones como

```
POST graph.facebook.com/{ user-id}/feed?message={message}&access_token={access-token}
```

Por completitud incluimos a continuación el código entero de la actividad.

```
package org.jordi.pruebafacebook2017sdk1;

//-----
// imports
//-----

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import com.facebook.AccessToken;
import com.facebook.CallbackManager;
import com.facebook.FacebookCallback;
import com.facebook.FacebookException;
import com.facebook.FacebookSdk;
import com.facebook.GraphRequest;
import com.facebook.GraphResponse;
import com.facebook.HttpMethod;
import com.facebook.Profile;
import com.facebook.login.LoginManager;
import com.facebook.login.LoginResult;
import com.facebook.login.widget.LoginButton;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.util.Log;
import android.view.View;
import android.view.inputmethod.InputMethodManager;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import org.json.JSONObject;

import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.util.Arrays;

import com.facebook.FacebookSdk;

//-----
// class MainActivity
//-----
public class MainActivity extends AppCompatActivity {
```




```
// -----  
// elementos gráficos  
// -----  
private TextView elTextoDeBienvenida;  
private Button botonHacerLogin;  
private Button botonLogout;  
private Button botonEnviarFoto;  
private TextView textoConElMensaje;  
private Button botonCompartir;  
  
//  
// boton oficial de Facebook para login/logout  
//  
LoginButton loginButtonOficial;  
  
//  
// gestiona los callbacks al FacebookSdk desde el método onActivityResult() de  
una actividad  
//  
private CallbackManager elCallbackManagerDeFacebook;  
  
//-----  
// puntero a this para los callback  
//-----  
private final Activity THIS = this;  
  
//-----  
// onCreate ()  
//-----  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    //  
    // inicializar super  
    //  
    super.onCreate(savedInstanceState);  
  
    //  
    Log.d("cuandrav.onCreate()", " .onCreate() llamado");  
  
    //  
    // cosas de Facebook  
    //  
  
    //  
    // inicializar FacebookSDK  
    // 2017: no hace falta, se puede borrar:  
    http://stackoverflow.com/questions/41904350/facebook-sdk-initialize-getapplicationcontext-deprecated  
    //  
    // FacebookSdk.sdkInitialize(this.getApplicationContext());  
  
    //  
    // pongo el contenido visual de la actividad (hacer antes que findViewById ())  
    // y después de inicializar FacebookSDK  
    //  
    this.setContentView(R.layout.activity_main);  
  
    //  
    // botón oficial de "login en Facebook"
```



```
//

// obtengo referencia
loginButtonOficial = (LoginButton) findViewById(R.id.Login_button);

// declaro los permisos que debe pedir al ser pulsado
// ver lista en: https://developers.facebook.com/docs/facebook-login/permissions
loginButtonOficial.setPublishPermissions("publish_actions");

//loginButtonOficial.setReadPermissions("public_profile"); // si pones uno,
no puedes poner el otro

//
// crear callback manager de Facebook
//
this.elCallbackManagerDeFacebook = CallbackManager.Factory.create();

// registro un callback para saber cómo ha ido el login
LoginManager.getInstance().registerCallback(this.elCallbackManagerDeFacebook,
    new FacebookCallback<LoginResult>() {
        @Override
        public void onSuccess(LoginResult loginResult) {
            // App code
            Toast.makeText(THIS, "Login onSuccess()",
                Toast.LENGTH_LONG).show();
            actualizarVentanita();
        }

        @Override
        public void onCancel() {
            Toast.makeText(THIS, "Login onCancel()",
                Toast.LENGTH_LONG).show();
            actualizarVentanita();
        }

        @Override
        public void onError(FacebookException exception) {
            // App code
            Toast.makeText(THIS, "Login onError(): " +
                exception.getMessage(),
                Toast.LENGTH_LONG).show();
            actualizarVentanita();
        }
    });

//
// otras cosas
//

//
// obtengo referencias a mis otros widgets en el layout
//
elTextoDeBienvenida = (TextView) findViewById(R.id.elTextoDeBienvenida);
botonHacerLogin = (Button) findViewById(R.id.boton_hacerLogin);
botonLogOut = (Button) findViewById(R.id.boton_LogOut);
botonEnviarFoto = (Button) findViewById(R.id.boton_EnviarFoto);
textoConElMensaje = (TextView) findViewById(R.id.txt_mensajeFB);
botonCompartir = (Button) findViewById(R.id.boton_EnviarAFB);

//
//
//
this.actualizarVentanita();
```



```
//
Log.d("cuandrav.onCreate", "final .onCreate() ");

} // ()

// -----
// -----
@Override
protected void onActivityResult(final int requestCode, final int resultCode,
final Intent data) {
    // se llama cuando otra actividad que hemos arrancado termina y nos devuelve
    el control
    // tal vez, devolviéndonos algun resultado (resultCode, data)
    Log.d("cuandrav.onActivityResult", "llamado");

    //
    // avisar a super
    //
    super.onActivityResult(requestCode, resultCode, data);

    //
    // avisar a Facebook (a su callback manager) por si le afecta
    //
    this.elCallbackManagerDeFacebook.onActivityResult(requestCode, resultCode,
data);
}

// -----
// -----
private void actualizarVentanita() {

    Log.d("cuandrav.actualizarVent", "empiezo");
    //
    // obtengo el access token para ver si hay sesión
    //
    AccessToken accessToken = this.obtenerAccessToken();

    if (accessToken == null) {

        Log.d("cuandrav.actualizarVent", "no hay sesion, deshabilito");
        //
        // sesion con facebook cerrada
        //
        this.botonHacerLogin.setEnabled(true);
        this.botonLogout.setEnabled(false);
        this.textoConElMensaje.setEnabled(false);
        this.botonCompartir.setEnabled(false);
        this.botonEnviarFoto.setEnabled(false);
        this.elTextoDeBienvenida.setText("haz login");

        return;
    }

    //
    // sí hay sesión
    //
    Log.d("cuandrav.actualizarVent", "hay sesion habilito");

    this.botonHacerLogin.setEnabled(false);
    this.botonLogout.setEnabled(true);
    this.textoConElMensaje.setEnabled(true);
}
```



```
this.botonCompartir.setEnabled(true);
this.botonEnviarFoto.setEnabled(true);

//
// averiguo los datos básicos del usuario acreditado
//
Profile profile = Profile.getCurrentProfile();
if (profile != null) {
    this.textoConElMensaje.setText(profile.getName());
}

//
// otra forma de averiguar los datos básicos:
// hago una petición con "graph api" para obtener datos del usuario
acreditado
//
this.obtenerPublicProfileConRequest_async(
    // como es asíncrono he de dar un callback
    new GraphRequest.GraphJSONObjectCallback() {
        @Override
        public void onCompleted(JSONObject datosJSON, GraphResponse
response) {
            //
            // muestro los datos
            //
            String nombre= "nombre desconocido";
            try{
                nombre = datosJSON.getString("name");
            } catch (org.json.JSONException ex) {
                Log.d("cuandrav.actualizarVent", "callback de
obtenerPublicProfileConRequest_async: excepcion: "
                    + ex.getMessage());
            } catch (NullPointerException ex) {
                Log.d("cuandrav.actualizarVent", "callback de
obtenerPublicProfileConRequest_async: excepcion: "
                    + ex.getMessage());
            }
            elTextoDeBienvenida.setText("bienvenido 2017: " + nombre);
        }
    });

} // ()

// -----
// -----
private boolean sePuedePublicar() {
    //
    // compruebo la red
    //
    if (!this.hayRed()) {
        Toast.makeText(this, "¿no hay red? No puedo publicar",
Toast.LENGTH_LONG).show();
        return false;
    }

    //
    // compruebo permisos
    //
    if (! this.tengoPermisoParaPublicar()) {
        Toast.makeText(this, "¿no tengo permisos para publicar? Los pido.",
Toast.LENGTH_LONG).show();
    }
}
```



```
// pedirPermisoParaPublicar();
LoginManager.getInstance().loginWithPublishPermissions(this,
Arrays.asList("publish_actions"));

    return false;
}

return true;
}

// -----
// -----
private AccessToken obtenerAccessToken() {
    return AccessToken.getCurrentAccessToken();
}

// -----
// -----
private boolean tengoPermisoParaPublicar() {
    AccessToken accessToken = this.obtenerAccessToken();
    return accessToken != null &&
accessToken.getPermissions().contains("publish_actions");
}

// -----
// -----
private boolean hayRed() {
    ConnectivityManager connectivityManager = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
    NetworkInfo activeNetworkInfo = connectivityManager
.getActiveNetworkInfo();
    return activeNetworkInfo != null && activeNetworkInfo.isConnected();

    // http://stackoverflow.com/questions/15091591/post-on-facebook-wall-without-showing-dialog-on-android
    // comprobar que estamos conetados a internet, antes de hacer el login con
    // facebook. Si no: da problemas.
} // ()

// -----
// -----
public void enviarTextoAFacebook_async (final String textoQueEnviar) {
    //
    // si no se puede publicar no hago nada
    //
    if ( ! sePuedePublicar() ) {
        return;
    }

    //
    // hago la petición a través del API Graph
    //
    Bundle params = new Bundle();
    params.putString("message", textoQueEnviar);

    GraphRequest request = new GraphRequest(
        AccessToken.getCurrentAccessToken(),
        "/me/feed",
        params,
        HttpMethod.POST,
        new GraphRequest.Callback() {
            public void onCompleted(GraphResponse response) {
```



```
        Toast.makeText(THIS, "Publicación realizada: " +
textoQueEnviar, Toast.LENGTH_LONG).show();

    }

};

request.executeAsync();

} // ()

// -----
// -----
public void enviarFotoAFacebook_async (Bitmap image, String comentario) {

    Log.d("cuandrav.envFotoFBasync", "llamado");

    if (image == null){

        Toast.makeText(this, "Enviar foto: la imagen está vacía.",
Toast.LENGTH_LONG).show();

        Log.d("cuandrav.envFotoFBasync", "acabo porque la imagen es null");
        return;
    }

    //
    // si no se puede publicar no hago nada
    //
    if ( ! sePuedePublicar() ) {
        return;
    }

    //
    // convierto el bitmap a array de bytes
    //
    ByteArrayOutputStream stream = new ByteArrayOutputStream();
    image.compress(Bitmap.CompressFormat.PNG, 100, stream);
    //image.recycle ();
    final byte[] byteArray = stream.toByteArray();
    try {
        stream.close();
    } catch (IOException e) { }

    //
    // hago la petición a través del Graph API
    //
    Bundle params = new Bundle();
    params.putByteArray("source", byteArray); // bytes de la imagen
    params.putString("caption", comentario); // comentario

    // si se quisiera publicar una imagen de internet:  params.putString("url",
"{image-url}");

    GraphRequest request = new GraphRequest(
        AccessToken.getCurrentAccessToken(),
        "/me/photos",
        params,
        HttpMethod.POST,
        new GraphRequest.Callback() {
            public void onCompleted(GraphResponse response) {
```



```
        Toast.makeText(THIS, "" + byteArray.length + " Foto enviada: " + response.toString(), Toast.LENGTH_LONG).show();
        //textoConElMensaje.setText(response.toString());
    }
}

);

request.executeAsync();

} // ()

// -----
// -----
public void boton_Login_pulsado(View quien) {
    //
    // compruebo la red
    //
    if (!this.hayRed()) {
        Toast.makeText(this, "¿No hay red? No puedo abrir sesión",
            Toast.LENGTH_LONG).show();
    }

    //
    // login
    //
    LoginManager.getInstance().loginWithPublishPermissions(this,
        Arrays.asList("publish_actions"));

    //
    // actualizar
    //
    this.actualizarVentanita();
} // ()

// -----
// -----
public void boton_Logout_pulsado(View quien) {
    //
    // compruebo la red
    //
    if (!this.hayRed()) {
        Toast.makeText(this, "¿No hay red? No puedo cerrar sesión",
            Toast.LENGTH_LONG).show();
    }

    //
    // logout
    //
    LoginManager.getInstance().logout();

    //
    // actualizar
    //
    this.actualizarVentanita();
} // ()

// -----
// -----
private void obtenerPublicProfileConRequest_async
(GraphRequest.GraphJSONObjectCallback callback) {
```




```
if (!this.hayRed()) {
    Toast.makeText(this, "¿No hay red?", Toast.LENGTH_LONG).show();
}

//
// obtengo access token y compruebo que hay sesión
//
AccessToken accessToken = obtenerAccessToken();
if (accessToken == null) {
    Toast.makeText(THIS, "no hay sesión con Facebook",
Toast.LENGTH_LONG).show();
    return;
}

//
// monto la petición: /me
//
GraphRequest request = GraphRequest.newMeRequest(accessToken, callback);
Bundle params = new Bundle ();
params.putString("fields", "id, name");
request.setParameters(params);

//
// la ejecuto (asíncronamente)
//
request.executeAsync();
}

// -----
// -----
public void boton_enviarTextoAFB_pulsado(View quien) {
    //
    // cojo el mensaje que ha escrito el usuario
    //

    String mensaje = "msg:" + this.textoConElMensaje.getText() + " : "
        + System.currentTimeMillis();

    //
    // borro lo escrito
    //
    this.textoConElMensaje.setText("");

    //
    // cierro el soft-teclado
    //
    InputMethodManager imm = (InputMethodManager)
getSystemService(Context.INPUT_METHOD_SERVICE);
    imm.hideSoftInputFromWindow(this.textoConElMensaje.getWindowToken(), 0);

    //
    // llamo al método que publica
    //
    enviarTextoAFacebook_async(mensaje);
}

} // class
```



8.1.4. Aplicación de ejemplo (Share Dialog)

Veamos ahora un pequeño ejemplo sobre cómo utilizar la funcionalidad de “Share Dialog” para publicar en Facebook indirectamente (a través de la app oficial de Facebook) y sin hacer login desde nuestra propia aplicación.

Configuraremos nuestra aplicación como hemos explicado antes. Damos de alta nuestra aplicación en Facebook developers y copiamos allí el keyhash y el nombre de la clase de nuestra actividad. Luego, en strings.xml definiremos el valor “facebook_app_id” con el número de aplicación que Facebook nos ha dado.

```
<string name="facebook_app_id">874076352708325</string>
```

En AndroidManifest.xml realizaremos cuatro modificaciones de forma que éste quede como a continuación. Especialmente, remarcamos que para enviar fotos con Share Dialog, hay que añadir un aviso de “content provider” con un nombre que incluye el ID de nuestra aplicación

com.facebook.app.FacebookContentProvider874076352708325

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="org.cuandrav.pruebafacebooksdk2">

    <!-- 1. añadir estos dos permisos -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <!-- 2. añadir por Facebook -->
        <meta-data android:name="com.facebook.sdk.ApplicationId"
            android:value="@string/facebook_app_id" />

        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

        <!-- 3. añadir por Facebook -->
        <activity android:name="com.facebook.FacebookActivity" />

        <!-- 4. añadir por Facebook -->
        <!-- para poder enviar fotos con Share Dialog. Fijaos dónde hay que poner el
        Facebook App ID -->
        <!-- ver: https://developers.facebook.com/docs/android/getting-started,
        sección Enviar Mensajes o Videos Sending Images or Videos -->
        <provider
            android:authorities="com.facebook.app.FacebookContentProvider874076352708325"
            android:name="com.facebook.FacebookContentProvider"
            android:exported="true" />
    </application>

</manifest>
```



Android: Programación Avanzada

Sobre el código de la actividad podemos comentar los siguientes aspectos:

- Crearemos un objeto “ShareDialog” en “onCreate()”.

```
this.elShareDialog = new ShareDialog(this);
```

- Disponemos del método “canShow()” para interrogar al share dialog si es capaz de realizar una determinada acción (publicar, publicar foto, publicar enlace, publicar video, ...) Si tenemos instalada la app oficial de Facebook para Android, la respuesta será “verdadero”. En caso que fuera falso, algunas las acciones se pueden realizar igualmente de forma externa a través de una página web que la biblioteca ya se encarga de mostrar.

```
boolean respuesta = ShareDialog.canShow(ShareLinkContent.class);
```

```
boolean respuesta = ShareDialog.canShow(SharePhotoContent.class);
```

- Para realizar una publicación el método siempre es el mismo, la única diferencia es la definición del objeto “content” que representa aquello que queremos publicar.

```
this.elShareDialog.show(content);
```

Esto abrirá una actividad diferente de la nuestra en la que podremos escribir y enviar a Facebook:



- Si queremos publicar en Facebook un enlace a una página web que nos ha gustado crearemos un objeto “content” utilizando “ShareLinkContent.Builder”

```
ShareLinkContent content = new ShareLinkContent.Builder()  
    .setContentUrl(Uri.parse("https://developers.facebook.com"))  
    .setContentDescription("me ha gustado esta página")  
    .build();
```

- Si queremos publicar un mensaje podemos utilizar el ShareLinkContent sin especificar el enlace URL.

```
ShareLinkContent content = new ShareLinkContent.Builder().build();
```

- Si queremos publicar una foto utilizaremos “SharePhoto.builder”



```
Bitmap Image =SharePhoto photo = new SharePhoto.Builder()  
    .setBitmap(image).build();  
SharePhotoContent content = new SharePhotoContent.Builder()  
    .addPhoto(photo).build();
```

Por completitud, reproducimos el código completo de esta activity que utiliza “Share Dialog”.

```
package org.jordi.pruebafacebook2017sdk2;  
  
import ...  
  
//-----  
// class MainActivity  
//-----  
public class MainActivity extends AppCompatActivity {  
  
    //  
    // elementos graficos  
    //  
  
    private Button boton2;  
    private Button boton3;  
  
    private Button boton1;  
    private TextView textoEntrada1;  
    private TextView textoSalida1;  
  
    //  
    // gestiona los callbacks al FacebookSdk desde el método onActivityResult() de  
    // una actividad  
    //  
    private CallbackManager elCallbackManagerDeFacebook;  
  
    //  
    // shareDialog  
    //  
    private ShareDialog elShareDialog;  
  
    //  
    // puntero a this para los callback  
    //  
    private final Activity THIS = this;  
  
    // -----  
    // -----  
    private void conseguirReferenciasAElementosGraficos () {  
        boton1 = (Button) findViewById(R.id.boton1);  
        boton2 = (Button) findViewById(R.id.boton2);  
        boton3 = (Button) findViewById(R.id.boton3);  
  
        textoEntrada1 = (TextView) findViewById(R.id.textoEntrada1);  
        textoSalida1 = (TextView) findViewById(R.id.textoSalida1);  
    } // ()  
  
    // -----  
    // -----  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {
```



```
//
// inicializar super
//
super.onCreate(savedInstanceState);

//
//
Log.d("cuandrav.onCreate()", " .onCreate() llamado");

//
//
// inicializar FacebookSDK
//
FacebookSdk.sdkInitialize(this.getApplicationContext());

//
// pongo el contenido visual de la actividad (hacer antes que findViewById ())
// y después de inicializar FacebookSDK
//
this.setContentView(R.layout.activity_main);

conseguirReferenciasAElementosGraficos();

//
// crear callback manager de Facebook
//
this.elCallbackManagerDeFacebook = CallbackManager.Factory.create();

//
// crear objeto share dialog
//
this.elShareDialog = new ShareDialog(this);

this.elShareDialog.registerCallback(this.elCallbackManagerDeFacebook, new
FacebookCallback<Sharer.Result>() {

    @Override
    public void onSuccess(Sharer.Result result) {
        Toast.makeText(THIS, "Sharer onSuccess()", Toast.LENGTH_LONG).show();
    }

    @Override
    public void onCancel() {}

    @Override
    public void onError(FacebookException error) {
        Toast.makeText(THIS, "Sharer onError(): " + error.toString(),
Toast.LENGTH_LONG).show();
    }

});

}

// -----
// -----
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    this.elCallbackManagerDeFacebook.onActivityResult(requestCode, resultCode,
data);
}

// -----
```



```
// -----  
private void publicarMensajeConIntent () {  
    Intent shareIntent = new Intent();  
    shareIntent.setAction(Intent.ACTION_SEND);  
  
    startActivityForResult(Intent.createChooser(shareIntent, "Share"), 1234);  
//requestId);  
} //  
  
// -----  
// -----  
public void boton1_pulsado(View quien) {  
    Log.d("cuandrav.boton1_pulsado", " llamado ");  
    textoSalida1.setText("boton1_pulsado");  
    this.publicarMensajeConIntent();  
} // ()  
  
// -----  
// -----  
private boolean puedoUtilizarShareDialogParaPublicarMensaje () {  
    return puedoUtilizarShareDialogParaPublicarLink();  
}  
  
// -----  
private boolean puedoUtilizarShareDialogParaPublicarLink () {  
    return ShareDialog.canShow(ShareLinkContent.class);  
} // ()  
  
// -----  
// -----  
private boolean puedoUtilizarShareDialogParaPublicarFoto () {  
    return ShareDialog.canShow(SharePhotoContent.class);  
} // ()  
  
// -----  
// -----  
public void boton2_pulsado(View quien) {  
    Log.d("cuandrav.boton2_pulsado", " llamado ");  
  
    textoSalida1.setText("boton2_pulsado");  
  
    //  
    // llamar al metodo para publicar  
    //  
    this.publicarMensajeConShareDialog();  
} // ()  
  
// -----  
// -----  
private void publicarMensajeConShareDialog () {  
  
    // https://developers.facebook.com/docs/android/share -> Using the Share  
Dialog  
  
    if ( ! puedoUtilizarShareDialogParaPublicarMensaje() ) {  
        Log.d("cuandrav.boton2_pul()", " ¡¡¡ No puedo utilizar share dialog  
!!!");  
  
        return;  
    }  
}
```



```
//  
// llamar a share dialog  
// aunque utilizamos ShareLinkContent, al no poner link  
// publica un mensaje  
//  
  
ShareLinkContent content = new ShareLinkContent.Builder().build();  
  
this.elShareDialog.show(content);  
  
} // ()  
  
// -----  
public void boton3_pulsado(View quien) {  
    Log.d("cuandrav.boton3_pulsado", " llamado ");  
  
    textoSalida1.setText("boton3_pulsado");  
  
    //  
    // llamar al metodo para publicar foto  
    //  
    this.publicarFotoConShareDialog();  
} // ()  
  
// -----  
// -----  
private void publicarFotoConShareDialog () {  
  
    // https://developers.facebook.com/docs/android/share -> Using the Share  
Dialog  
  
    if ( ! puedoUtilizarShareDialogParaPublicarFoto() ) {  
        return;  
    }  
  
    //  
    // cojo una imagen directamente de los recursos  
    // para publicarla  
    //  
    Bitmap image = BitmapFactory.decodeResource(  
        getResources(), R.drawable.com_facebook_logo);  
  
    //  
    // monto la petición  
    //  
    SharePhoto photo = new SharePhoto.Builder()  
        .setBitmap(image)  
        .build();  
    SharePhotoContent content = new SharePhotoContent.Builder()  
        .addPhoto(photo)  
        .build();  
  
    this.elShareDialog.show(content);  
  
} // ()  
  
} // class
```




Práctica:

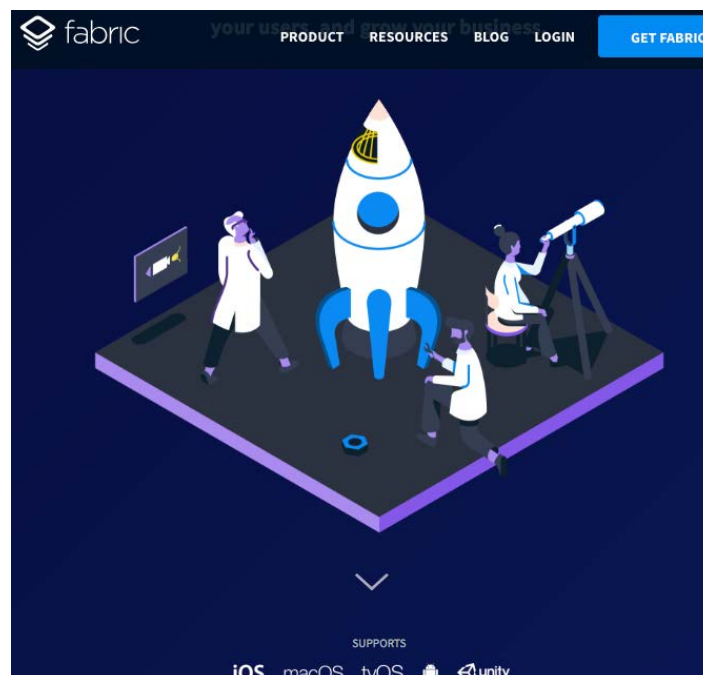
Escribe una aplicación que pueda publicar mensajes y fotos en Facebook, utilizando los ejemplos anteriores. (Previamente hay que haber realizado el ejercicio para dar de alta la aplicación en Facebook developers).



Preguntas de repaso: [Facebook](#)

8.2. Android y Twitter

Recientemente, Twitter ha desarrollado un conjunto de utilidades para integrar aplicaciones móviles con su plataforma, con el nombre de [Fabric](#). Esta plataforma ha sido adquirida por Google.

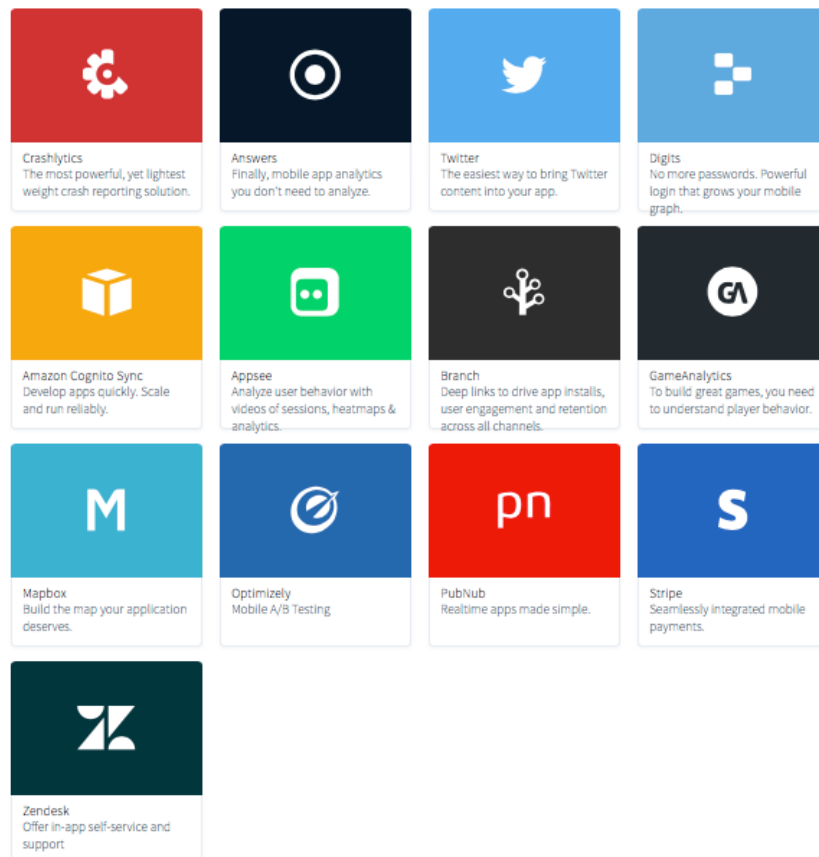


Una de las utilidades más destacadas es [Crashlytics \(www.crashlytics.com\)](http://www.crashlytics.com), una herramienta para generar informes de errores y fallos en aplicaciones móviles, facilitando su corrección. Pero hay otros muchos [kits \(fabric.io/kits\)](http://fabric.io/kits) disponibles:

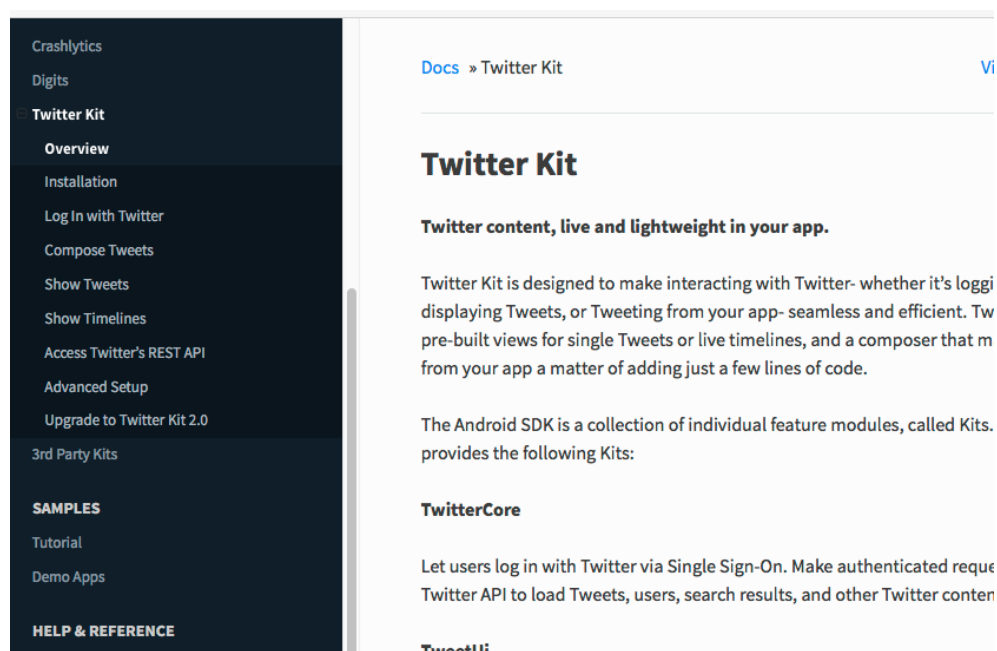


Android: Programación Avanzada

Jump right to the kit you need



incluyendo mapas, pagos, almacenamiento en la nube, acreditación de usuarios, y por supuesto la posibilidad de integrar Twitter en nuestra aplicación mediante [TwitterKit \(fabric.io/kits/android/twitterkit\)](https://fabric.io/kits/android/twitterkit). Para utilizar esta plataforma, seguiremos la guía oficial: docs.fabric.io/android/twitter/overview.html

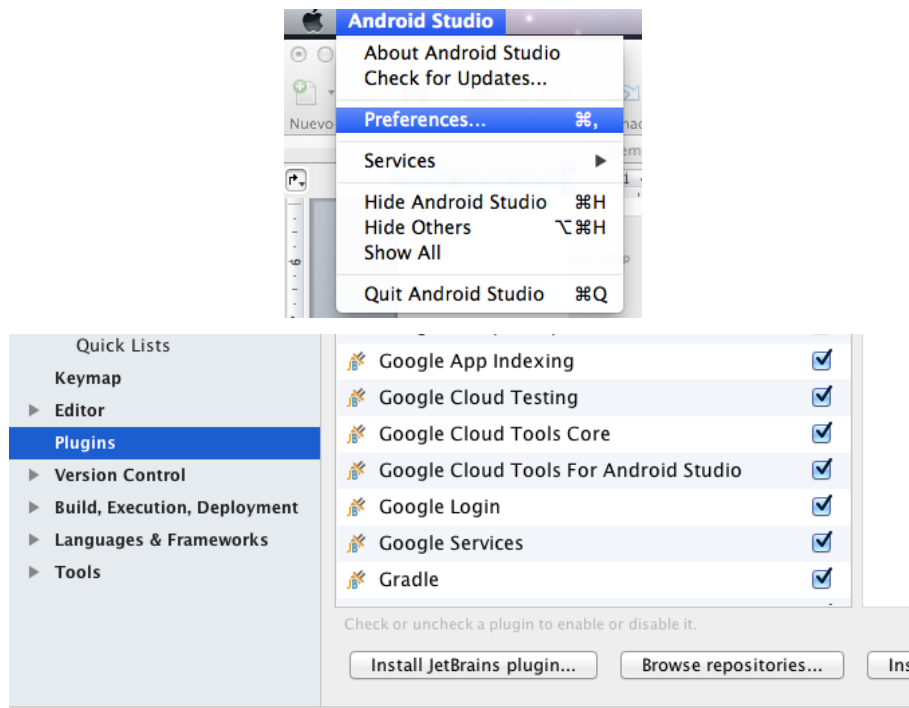




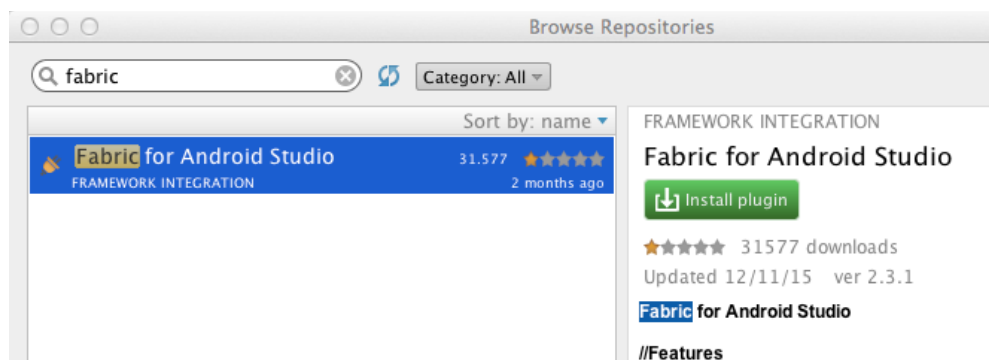
Android: Programación Avanzada

8.2.1. Instalando Fabric en Android Studio

Empezaremos por instalar el plugin de Fabric en Android Studio. Vamos a preferencias de Android Studio.



Allí, elegimos “Plugins” y pulsamos el botón “Browse repositories”, lo cual nos llevará a un diálogo donde escribimos “Fabric” en el texto de búsqueda para encontrar el plugin que instalaremos pulsando el botón verde “Install plugin” (necesitaremos reiniciar Android Studio para activarlo).



Cuando el plugin está instalado, en el margen izquierdo de la ventana de Android Studio veremos un botón con el logo de Fabric en Android Studio.



Android: Programación Avanzada



Al pulsar dicho botón, se muestra la ventana de Fabric .



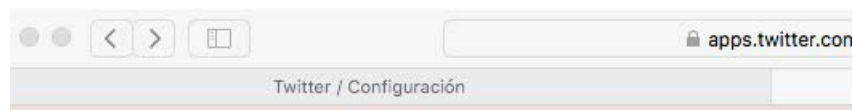
Ejercicio: Instala el plugin de “fabric” en Android Studio.

Nota: Alternativamente a utilizar el plugin, el kit de fabric para Twitter se puede instalar modificando ficheros de gradle, tal y como se explica aquí: fabric.io/kits/android/twitterkit/install

8.2.2. Configurando nuestra aplicación en Twitter Developers

Cada aplicación de Android integrada con Twitter tiene que estar dada de alta en esta red social. Para ellos vamos a

<https://apps.twitter.com> y pulsamos en el botón “Create New App”. En la pantalla que aparece, rellenamos los 4 campos: nombre, descripción, sitio web y callback URL. Es importante rellenar también el callback, aunque sea con un url ficticio como <http://www.example.com>, ya que esto va a permitir volver a nuestra aplicación de forma transparente desde la página web de Twitter en donde nos autenticaremos.



Create an application

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authentication screens.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 1 and 140 characters.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or get more information about your application. A fully qualified URL is used in the source attribution for tweets created by your application and will be visible to users. (If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should expect a redirect to this URL. To restrict your application from using callbacks, leave this field blank.

Ahora vamos a la pestaña "Permissions" de nuestra aplicación:

PruebaTwitter2017

[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)

Access

What type of access does your application need?

Read more about our [Application Permission Model](#).

- ☐ Read only
- ☒ Read and Write
- ☐ Read, Write and Access direct messages

Note:

Changes to the application permission model will only reflect in access tokens created after the change. Existing access tokens will need to be re-negotiated to alter the permissions.




Android: Programación Avanzada

y marcamos “Read and Write”, pulsando luego el botón “Update settings”. En la pestaña “Details” veremos algo como esto:

PruebaTwitter2017

[Details](#) [Settings](#) [Keys and Access Tokens](#) [Permissions](#)



prueba de aplicación android para twitter
<http://www.example.org>

Organization

Information about the organization or company associated with your application. This

Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to [authenticate](#) requests to the

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	Jpede8fbMirbYliDpMqzNtckM (manage keys)
Callback URL	http://www.example.org
Callback URL Locked	No
Sign in with Twitter	Yes
App-only authentication	https://api.twitter.com/oauth2/token

Luego, vamos a la pestaña “Keys and Access Tokens” para anotarnos “Consumer Key (API key)” y “Consumer Secret (API secret)”, que identifican a nuestra aplicación.



Prueba Twitter2017

Details

Settings

Keys and Access Tokens

Permissions

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your code.

Consumer Key (API Key)	Jpede8fbMlrBYIiDpMqzNtckM
Consumer Secret (API Secret)	AEd1vIR0SV3rGP1SLg8smEZNOFYEToNDw6
Access Level	Read and write (modify app permissions)
Owner	cuandrav
Owner ID	1091548429



Ejercicio: *Da de alta una aplicación en Twitter developers.*

8.2.3. Configurando Fabric en nuestra aplicación de Android Studio

En primer lugar, crearemos un proyecto nuevo en Android Studio. A continuación, pulsamos el botón "on" de Fabric.



Nos pregunta nuestro nombre de usuario (si no estamos registrados, podemos registrarnos con "Sign Up", utilizando una cuenta nuestra de Google, p.ej.).

Después, nos muestra los proyectos que tenemos dados de alta. Recordemos que toda aplicación que interactúe con Twitter (igual que ocurre en Facebook) debe estar dada de alta en la zona de desarrolladores de Twitter (ver sección anterior)

Si avanzamos dentro de la ventana de Fabric en Android Studio, veremos la lista de kits que podemos utilizar para desarrollo.



Android: Programación Avanzada

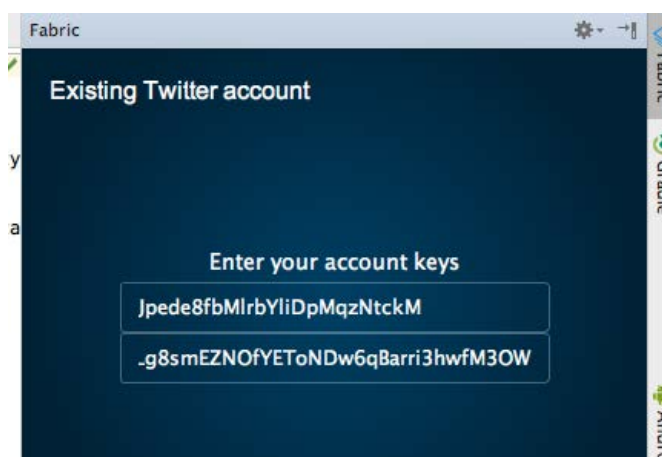


Nosotros elegiremos “Twitter” que nos lleva a una ventana donde vamos a instalar la biblioteca y desde donde podemos obtener código de ejemplo.



Para la instalación se requiere una cuenta en Twitter, que puede ser creada vinculándose al e-mail como desarrollador de Fabric que hemos creado o a una cuenta ya existente de Twitter (usuario convencional) que tuviéramos. En este segundo caso, elegiríamos “I already have a Twitter account”.

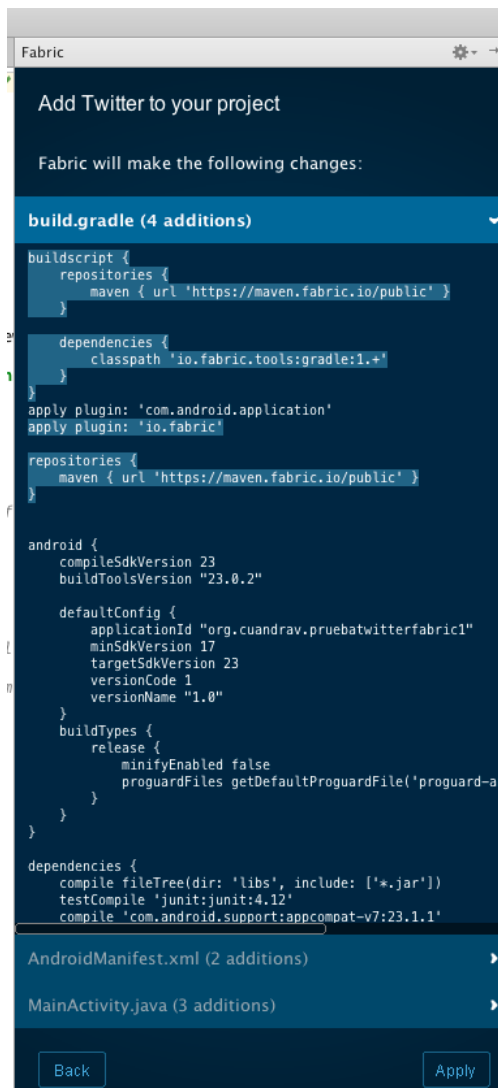
En la siguiente pantalla, debemos copiar el “API key” y el “API Secret” sacados de la pestaña “Keys and access tokens” de la configuración de nuestra aplicación (ver sección anterior).



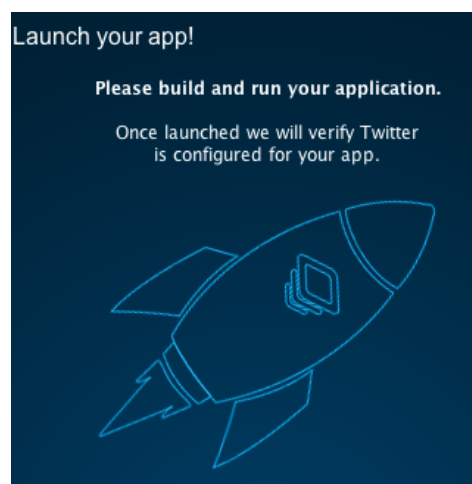
Ahora hay que hacer cambios en varios ficheros: build.gradle, AndroidManifest.xml, MainActivity.java. El plugin de Fabric nos permite realizarlos automáticamente pulsando “Apply”:



Android: Programación Avanzada



Al final de este proceso, en la ventana de Fabric veremos:



En relación a las claves de la aplicación, podemos ver que en MainActivity.java se ha añadido dos constantes:



Android: Programación Avanzada

```
private static final String TWITTER_KEY = "WOHF727DwGJBxy8uPtFJxA";
private static final String TWITTER_SECRET = "kc40JcVSuxZeWRgq";
```

con los valores antes especificados. Si hemos cometido algún error con las claves, aquí podemos cambiarlos.

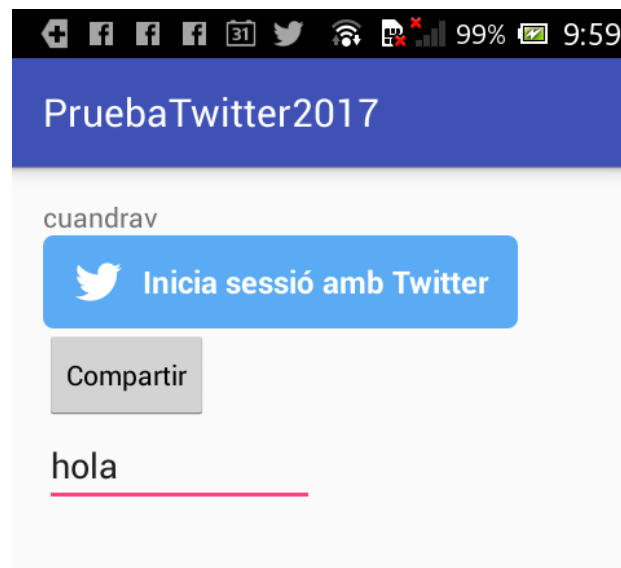


Ejercicio: Configura tu aplicación Twitter en Android Studio, siguiendo los anteriores pasos.

8.2.4. Aplicación de Ejemplo

Igual que ocurría con Facebook, advertimos que conviene tener la aplicación oficial de Twitter instalada en nuestro teléfono.

El código que vamos a estudiar como ejemplo simplemente sirve para hacer login en Twitter y publicar mensajes en él, en nombre de un usuario existente.



Siguiendo los pasos explicados en la sección anterior tendremos ya Android Studio y un proyecto preparado para utilizar Fabric. Revisemos, pues, los principales cambios que necesitamos en algunos de los ficheros de nuestro proyecto (algunos de los cuáles, el plugin de Fabric ya habrá realizado).

- En el AndroidManifest.xml habrá un nuevo metadato que referencia a Fabric (su "api key") y el permiso para acceder a internet.

```
<meta-data android:name="io.fabric.ApiKey"
            android:value="93e32ea1e8b8d08c046cbd05a67472cf054efc70" />
</application>

<uses-permission android:name="android.permission.INTERNET" />
```

- En el fichero build.gradle de la app se veremos en la sección de dependencias:

```
dependencies {
    compile 'com.squareup.retrofit:retrofit:1.9.0' // <-- añadido para usar esta
    biblioteca

    compile('com.twitter.sdk.android:twitter:2.3.2@aar') { // <-- Lo añade el plugin
        de fabric/twitter
    }
}
```



Android: Programación Avanzada

```
transitive = true;  
}
```

- En el layout de la actividad (activity_main.xml) debemos incluir este código para utilizar el botón oficial de Twitter:

```
<com.twitter.sdk.android.core.identity.TwitterLoginButton  
    android:id="@+id/twitter_login_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"/>
```

- Finalmente, en el código java de MainActivity.java tendremos:
 - La clave y contraseña que identifican a nuestra aplicación ante Twitter.

```
private static final String TWITTER_KEY = "WOHF727DwGJBxy8uPtFJxA";  
private static final String TWITTER_SECRET = "kc40JcVSux";
```

- En onCreate(), la inicialización de Fabric

```
TwitterAuthConfig authConfig = new TwitterAuthConfig(TWITTER_KEY, TWITTER_SECRET);  
Fabric.with(this, new Twitter(authConfig);
```

- En onCreate(), instalar un callback al botón oficial de Twitter para conocer el resultado del login. (Véase que podemos consultar datos del usuario autenticado en result.data).

```
botonEnviarATwitter = (Button) findViewById(R.id.boton_EnviarATwitter);  
  
botonLoginTwitter.setCallback(new Callback<TwitterSession>() {  
    @Override  
    public void success(Result<TwitterSession> result) {  
        Toast.makeText(THIS, "Autenticado en twitter: " + result.data.getUserName(),  
            Toast.LENGTH_LONG).show();  
    }  
  
    @Override  
    public void failure(TwitterException e) {  
        Toast.makeText(THIS, "Fallo en autenticación: " + e.getMessage(),  
            Toast.LENGTH_LONG).show();  
    }  
});
```

- Reescribir onActivityResult() para informar al botón oficial de Twitter (que es quien gestiona el proceso de login).

```
@Override  
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    botonLoginTwitter.onActivityResult(requestCode, resultCode, data);  
}
```

- Podemos publicar un tweet directamente desde nuestra aplicación a Twitter (via su interfaz REST) así:

```
StatusesService statusesService = Twitter.getApiClient( obtenerSesionDeTwitter() )  
    .getStatusesService();  
  
Call<Tweet> call = statusesService.update(textoQueEnviar, null, null, null, null,  
    null, null, null, null);  
  
call.enqueue(new Callback<Tweet>() {  
    @Override  
    public void success(Result<Tweet> result) {  
        Toast.makeText(THIS, "Tweet publicado: " + result.response.message(),  
            Toast.LENGTH_LONG).show();  
    }  
});
```



```
}  
@Override  
public void failure(TwitterException e) {  
    Toast.makeText(THIS, "No se pudo publicar el tweet: " + e.getMessage(),  
        Toast.LENGTH_LONG).show();  
}  
});
```

- También se puede publicar mediante la app oficial de Twitter o en su ausencia mediante página web externa a nuestra aplicación. En este caso, es sencillo incluir una imagen.

```
TweetComposer.Builder builder = new TweetComposer.Builder(this)  
    .text("esto es un tweet");  
//.image(myImageUri); // se puede añadir una imagen  
builder.show();
```

- Publicar una imagen directamente vía REST necesita bastante código. Para simplificar, recomendaríamos “esconderlo” en una función. Primero hay que copiar la imagen al Twitter. Después de este paso la imagen aún no se ve. Hay que publicar un tweet que la referencie. El siguiente es un código de ejemplo comentado.

```
File photo = null;  
try {  
    // 1. Abrimos el fichero con la imagen  
    // imagen que queremos enviar. Como necesitamos un path (para TypedFile)  
    // debe estar fuera de /res o /assets porque estos  
    // estan dentro del .apk y NO tiene path  
    photo = new File ("/storage/sdcard0/DCIM/100ANDRO/DSC_0001.jpg");  
  
} catch (Exception e) {  
    Log.d("miApp", "enviarImagen : excepcion: " + e.getMessage());  
    return;  
} // catch  
  
// 2. ponemos el fichero en un TypedFile  
  
TypedFile typedFile = new TypedFile("image/jpg", photo);  
  
// 3. obtenemos referencia al media service  
MediaService ms = Twitter.getApiClient( obtenerSesionDeTwitter()  
).getMediaService();  
  
// 3.1 ponemos la foto en el request body de la petición  
okhttp3.RequestBody requestBody = okhttp3.RequestBody.create(MediaType.parse  
("image/png"), photo);  
  
// 4. con el media service: enviamos la foto a Twitter  
Call<Media> call1 = ms.upload(  
    requestBody, // foto que enviamos  
    null,  
    null);  
  
call1.enqueue (new Callback<Media>() {  
    @Override  
    public void success(Result<Media> mediaResult) {  
        // he tenido éxito:  
        Toast.makeText(THIS, "imagen publicada: " +  
            mediaResult.response.toString(), Toast.LENGTH_LONG);  
  
        // 5. como he tenido éxito, la foto está en twitter, pero no en el  
        // timeline (no se ve) he de escribir un tweet referenciando la foto
```



```
// 6. obtengo referencia al status service
StatusesService statusesService =
TwitterCore.getInstance().getApiClient(obtenerSesionDeTwitter())
    .getStatusesService();

// 7. publico un tweet
Call<Tweet> call2 = statusesService.update("prueba de enviar imagen
"+System.currentTimeMillis() , // mensaje del tweet
    null,
    false,
    null,
    null,
    null,
    true,
    false,
    ""+mediaResult.data.mediaId // string con los identificadores
(hasta 4, separado por coma) de las imágenes
    // que quiero que aparezcan en este tweet. El mediaId
referencia a la foto que acabo de subir previamente
);

call2.enqueue(
    new Callback<Tweet>() {
        @Override
        public void success(Result<Tweet> result) {
            Toast.makeText(THIS, "Tweet publicado: "+
result.response.message().toString(), Toast.LENGTH_LONG).show();
        }
        @Override
        public void failure(TwitterException e) {
            Toast.makeText(THIS, "No se pudo publicar el tweet: "
"+ e.getMessage(), Toast.LENGTH_LONG).show();
        }
    });

}

@Override
public void failure(TwitterException e) {
    // failure de call1
    Toast.makeText(THIS, "No se pudo publicar el tweet: "
e.getMessage(), Toast.LENGTH_LONG).show();
}

});
```



Práctica: Escribe una aplicación que pueda enviar Tweets, utilizando los ejemplos anteriores.



Preguntas de repaso: [Twitter](#)