

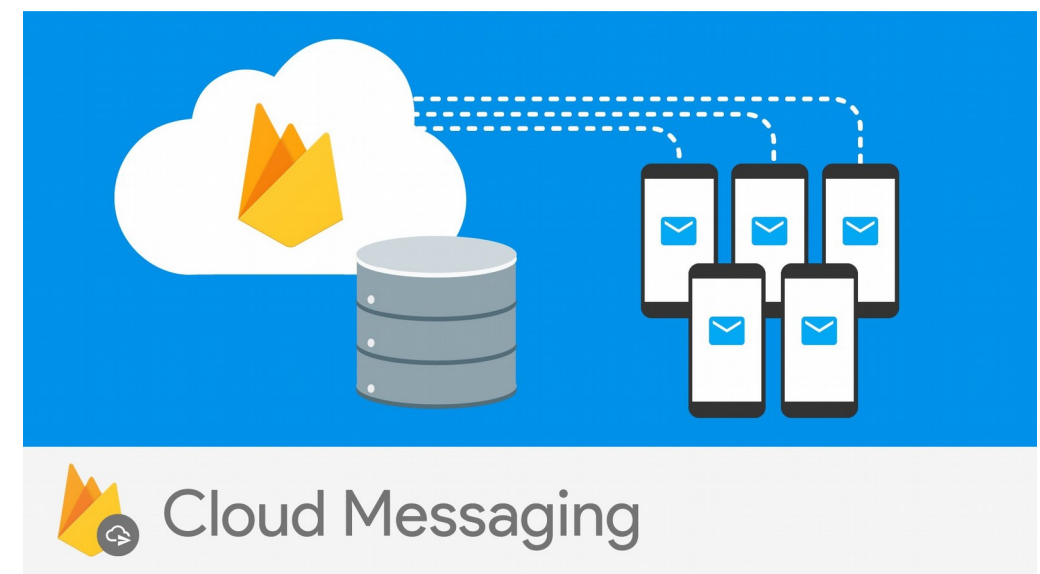


# Servicios de las plataformas móviles

## Sesión 1: Servicios de autenticación y notificaciones

## Puntos a tratar

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



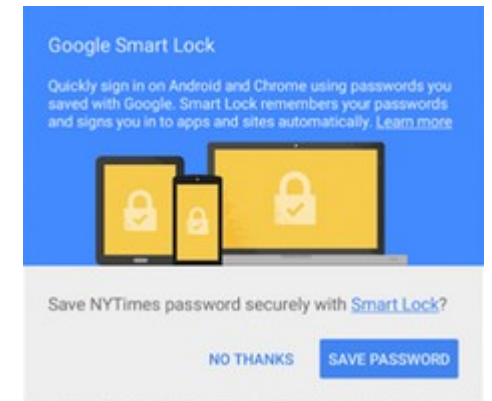


# Google Accounts



# Control de Acceso

- Autenticación
  - Google Sign-In
  - Smart Lock for Passwords
  - Firebase Authentication
  - OpenID Connect
  - Login con OAuth 2.0
- Autorización
  - OAuth 2.0





# Google Sing-In

- Requisitos de la plataforma Google Sign-In para Android
- Habilitación del servicio Google Sign-In
- Programación de la aplicación
  - Configuración del proyecto en Android Studio
  - Programación del proceso de identificación (GoogleApiClient)



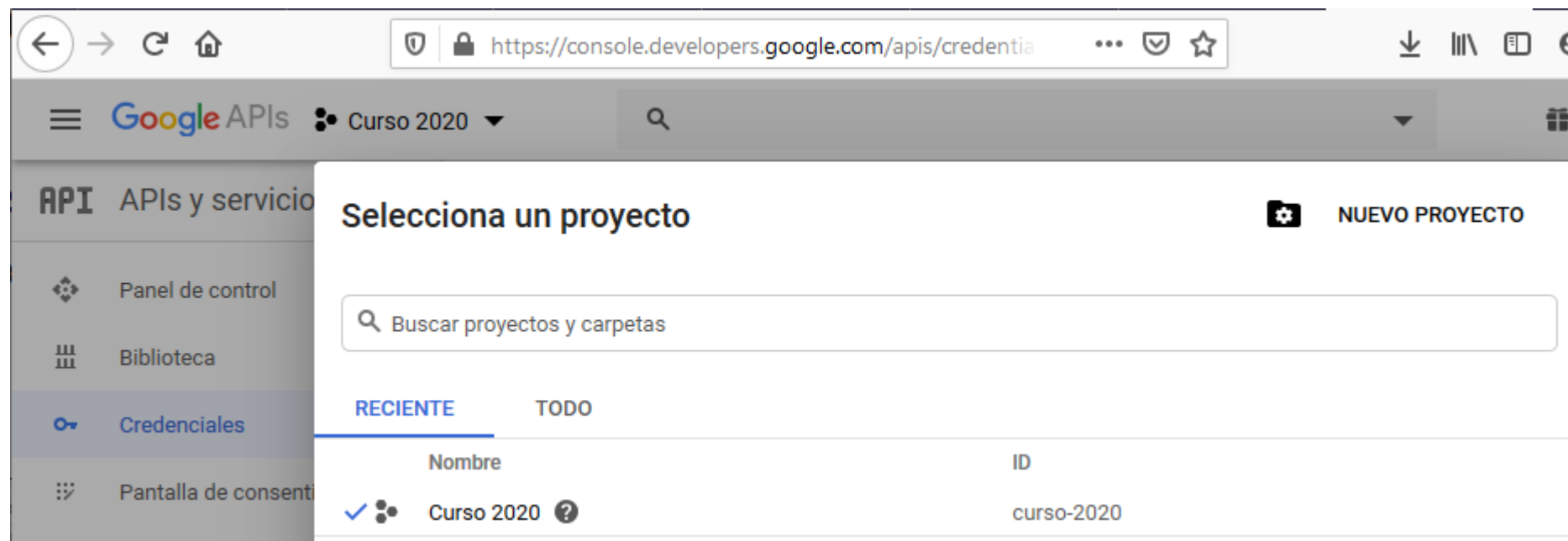
# Prerequisitos de la plataforma Google Sign-In para Android

- Un dispositivo compatible con Android que ejecute Android 4.1 o superior y que incluya el Google Play Store, o un emulador de AVD (Dispositivo Virtual de Android) que tenga las Google APIs de Android 4.2.2 o superior y tenga la versión 15.0.0 o superior de Google Play Services.
- La última versión del Android SDK, incluyendo el componente SDK Tools.
- Un proyecto configurado para compilar contra Android 4.1 (Jelly Bean) o superior.

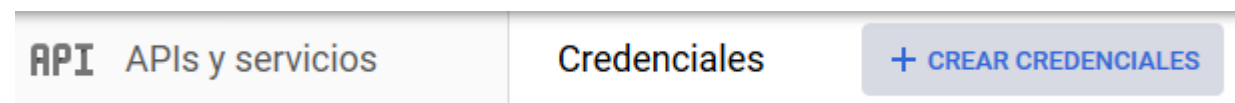


# Habilitación del servicio Google Sign-In

1. Ir a la dirección <https://console.developers.google.com>
2. Seleccionar un proyecto existente o crear uno nuevo.



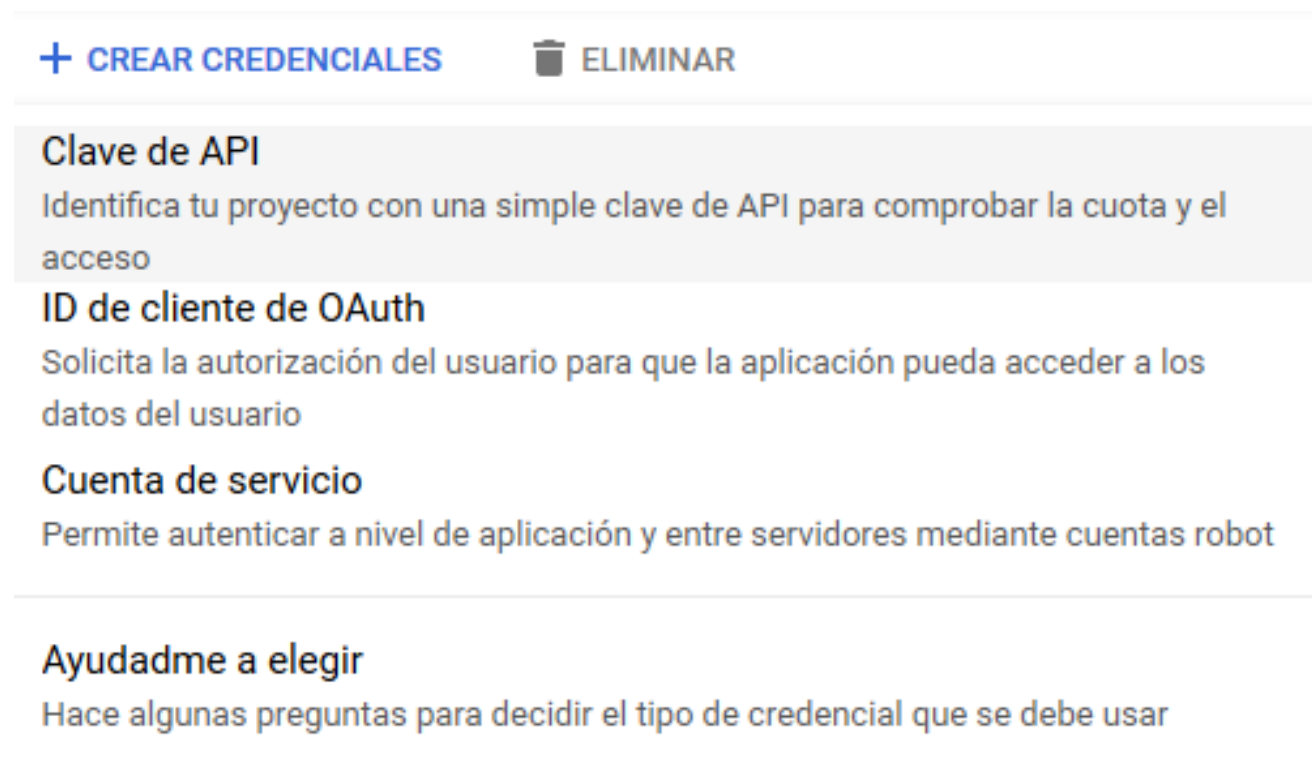
3. Pulsar el botón para crear unas credenciales nuevas para nuestra aplicación.





# Habilitación del servicio Google Sign-In mediante la consola de desarrolladores

4. Seleccionar el tipo de credencial. Seleccionamos “ID de cliente de OAuth”.







# Habilitación del servicio Google Sign-In mediante la consola de desarrolladores

5. Elegir el tipo de aplicación.  
Seleccionamos “Android”. Introducir la huella digital del certificado de firma e introducimos el nombre del paquete de la aplicación, así como un nombre para la credencial.

## Crear ID de cliente

### Tipo de aplicación

- ☐ Aplicación web  
☒ Android [Más información](#)  
☐ Aplicación de Chrome [Más información](#)  
☐ iOS [Más información](#)  
☐ PlayStation 4  
☐ Otro

### Nombre

Cliente de Android 1

### Huella digital de certificado de firma

Los dispositivos Android envían solicitudes de API directamente a Google. Google verifica que cada una de las solicitudes proceda de una aplicación para Android que coincida con un nombre de paquete y una huella digital de certificado de firma SHA-1 que nos proporciones. Usa el siguiente comando para obtener la huella digital. [Más información](#)

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

12:34:56:78:90:AB:CD:EF:12:34:56:78:90:AB:CD:EF:AA:BB:CC:DD

### Nombre del paquete

Del archivo AndroidManifest.xml

es.ua.gms

Crear

Cancelar



# Obtención huella del certificado (I)

## Desde un terminal

- Posicionarse en la carpeta en la que está la herramienta keytool (No necesario si la ruta está en la variable PATH).
- Ejecutar el comando:  
`keytool -exportcert -alias androiddebugkey -keystore %userprofile%/.android/debug.keystore -list -v`  
(ruta del almacén de certificados en windows)
- Introducir como contraseña **android**.
- La huella necesario es el valor que sale en **SHA1**

Certificate fingerprints:

MD5: 1B:2B:2D:37:E1:CE:06:8B:A0:F0:73:05:3C:A3:63:DD

SHA1: D8:AA:43:97:59:EE:C5:95:26:6A:07:EE:1C:37:8E:F4:F0:C8:05:C8

SHA256: F3:6F:98:51:9A:DF:C3:15:4E:48:4B:0F:91:E3:3C:6A:A0:97:DC:0A:3F:B2:D2

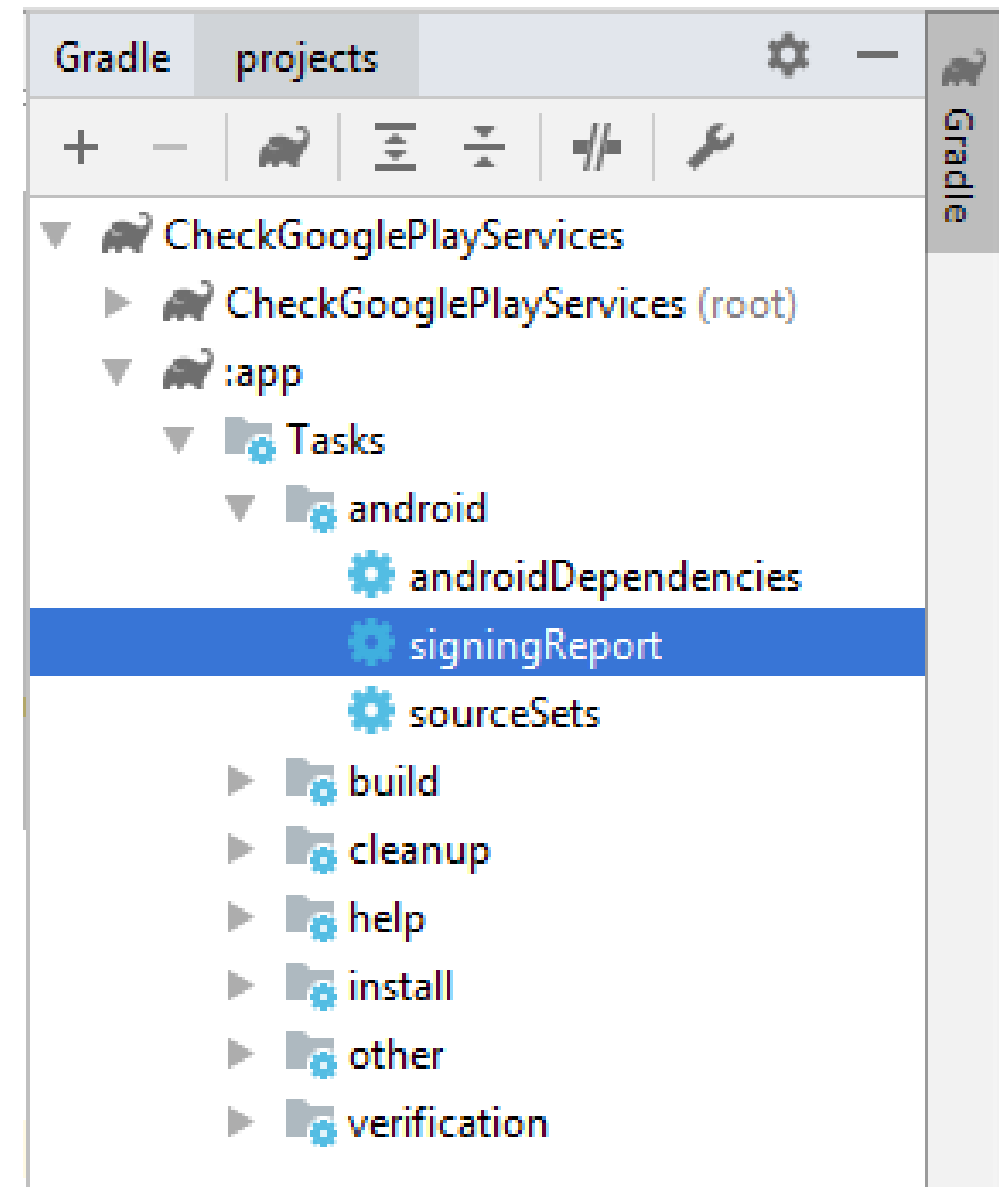
Signature algorithm name: SHA1withRSA

Version: 3

# Obtención huella del certificado (II)

## Desde Android Studio

- Pulsar en la pestaña de Gradle
- Pulsar el botón de refrescar proyectos
- Desplegar el proyecto raíz
- Desplegar la carpeta Tasks
- Desplegar la carpeta android
- Desplegar la carpeta androidDependencies
- Desplegar la carpeta signingReport
- Hacer DobleClick sobre signingReport
- La huella necesaria es el valor que sale en **SHA1**.





# Habilitación del servicio Google Sign-In mediante la consola de desarrolladores

## 5. Definir los datos de la pantalla de consentimiento de OAuth.

### Pantalla de consentimiento de OAuth

Elige cómo quieres configurar y registrar la aplicación, incluidos los usuarios objetivo. Solo puedes vincular una aplicación a tu proyecto.

#### User Type

☐ **Internos** ?

Solo estará disponible para los usuarios que pertenezcan a tu organización. No hará falta que envíes tu aplicación para verificarla.

☒ **Externo** ?

Estará a disposición de cualquier usuario con una cuenta de Google.

**CREAR**

### Pantalla de consentimiento de OAuth

En la pantalla de consentimiento, los usuarios deciden, antes de autenticarse, si quieren conceder acceso a sus datos privados. En ella también se incluyen enlaces a las condiciones del servicio y a la política de privacidad. En esta página se configura la pantalla de consentimiento de todas las aplicaciones del proyecto.

#### Estado de verificación

Sin publicar

#### Nombre de aplicación ?

Nombre de la aplicación que solicita el consentimiento

Curso 2020

#### Logotipo de la aplicación ?

Imagen que aparece en la pantalla de consentimiento para que los usuarios reconozcan la aplicación

Elige un archivo local para subirlo

**Examinar**



#### Correo electrónico de asistencia ?

Se muestra en la pantalla de consentimiento como contacto de asistencia

jabelda@gmail.com

#### Permisos para acceder a las API de Google

Los permisos sirven para que la aplicación acceda a los datos privados de los usuarios.

[Más información](#)

Si añades algún permiso sensible (por ejemplo, los que conceden acceso completo a Calendar o Drive), Google verificará tu pantalla de consentimiento antes de publicarla.

email

profile

openid



# Configuración del proyecto en Android Studio

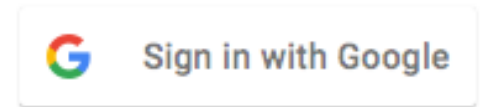
- De forma manual, escribimos lo siguiente en el fichero **build.gradle** del módulo

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

dependencies {
    implementation 'com.google.android.gms:play-services-auth:17.0.0'
}
```

# Programación del proceso de identificación

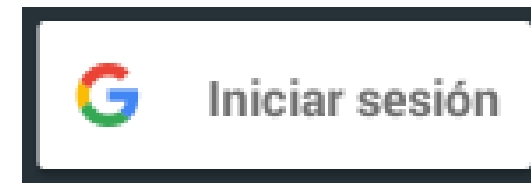
- Incluir el botón “Sign in with Google” (Opcional)
- Configurar las opciones de inicio de sesión (GoogleSignInOptions).
- Crear el objeto GoogleSignInClient.
- Comprobar si ya hay un usuario identificado.
- Iniciar el flujo de inicio de sesión.
- Obtener información del perfil.
- Finalización de la sesión.
- Desconexión de la cuenta.



# Añadir botón de Inicio de sesión

1. Añadir un *SignInButton* a nuestro layout.

```
<com.google.android.gms.common.SignInButton  
    android:id="@+id/sign_in_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content" />
```



2.(Opcional) Personalización del botón con `setSize` y/o `setColorScheme`.

```
// Set the dimensions of the sign-in button.  
SignInButton signInButton = findViewById(R.id.sign_in_button);  
signInButton.setSize(SignInButton.SIZE_STANDARD);
```

3.Registrar el *OnClickListener* del botón.

```
findViewById(R.id.sign_in_button).setOnClickListener(this);
```



# Configurar las opciones de Google Sign-In

- Lo hacemos en el método *onCreate* de la actividad de identificación.
- Utilizamos el objeto `GoogleSignInOptions`.
- Algunas opciones disponibles:
  - `requestID()`
  - `requestProfile()`
  - `GoogleSignInOptions.DEFAULT_SIGN_IN`
  - `requestEmail()`

```
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
    .requestEmail()
    .build();
```





# Creación del objeto GoogleSignInClient

- Lo hacemos en el método *onCreate* de la actividad de identificación también.
- Utilizamos las opciones definidas previamente.

```
// Build a GoogleSignInClient with the options specified by gso.  
mGoogleSignInClient = GoogleSignIn.getClient(this, gso);
```



# Comprobar si ya hay un usuario identificado

- Lo hacemos en el método *onStart* de la actividad de identificación.

```
// Check for existing Google Sign In account, if the user is already signed in  
// the GoogleSignInAccount will be non-null.  
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);  
updateUI(account);
```

Si hubiera un usuario ya identificado en la aplicación, `account` contendría un objeto **GoogleSignInAccount** con la cuenta del usuario identificado. De lo contrario, `account` tendría el valor **null**.

# Flujo de Inicio de sesión (I)

## 1. Gestionar la pulsación del botón de inicio de sesión.

1. Identificamos la pulsación del botón de inicio de sesión.
2. Creamos un intent de inicio de sesión con *getSignInIntent*.
3. Lanzamos el intent con *startActivityForResult*.

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        case R.id.sign_in_button:
            signIn();
            break;
        // ...
    }
}
```

```
private void signIn() {
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

## Flujo de Inicio de sesión (II)

### 2. Recuperar el resultado del intent de inicio de sesión.

1. Implementar en `onActivityResult` la identificación del intent de inicio de sesión.
2. Extraer en un objeto *GoogleSignInAccount* el resultado del intent.

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

    // Result returned from launching the Intent from GoogleSignInClient.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        // The Task returned from this call is always completed, no need to attach
        // a listener.
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        handleSignInResult(task);
    }
}
```



## Flujo de Inicio de sesión (III)

### 3. Gestionar el resultado de la identificación.

1. (Aconsejable) Definir un método privado para hacerlo.
2. Extraer de la tarea la cuenta que ha iniciado sesión.
3. Implementar la lógica del programa en cada caso.

```
private void handleSignInResult(Task<GoogleSignInAccount> completedTask) {  
    try {  
        GoogleSignInAccount account = completedTask.getResult(ApiException.class);  
  
        // Signed in successfully, show authenticated UI.  
        updateUI(account);  
    } catch (ApiException e) {  
        // The ApiException status code indicates the detailed failure reason.  
        // Please refer to the GoogleSignInStatusCodes class reference for more information.  
        Log.w(TAG, "signInResult:failed code=" + e.getStatusCode());  
        updateUI(null);  
    }  
}
```



## Obtener información del perfil

Una vez iniciada la sesión, si hemos configurado Google Sign-In con el parámetro *DEFAULT\_SIGN\_IN* o con el método *requestProfile*, podemos recuperar la información del perfil con *GoogleSignIn.getLastSignInAccount*.

Si también hemos utilizado el método *requestEmail* podemos recuperar la dirección de correo electrónico.

```
GoogleSignInAccount acct = GoogleSignIn.getLastSignedInAccount(getActivity());
if (acct != null) {
    String personName = acct.getDisplayName();
    String personGivenName = acct.getGivenName();
    String personFamilyName = acct.getFamilyName();
    String personEmail = acct.getEmail();
    String personId = acct.getId();
    Uri personPhoto = acct.getPhotoUrl();
}
```

# Cerrar sesión del usuario

Podemos añadir un botón para cerrar la sesión. Al pulsarlo llamaríamos al método *signOut* que solicitaría el final de la sesión.

```
@Override
public void onClick(View v) {
    switch (v.getId()) {
        // ...
        case R.id.button_sign_out:
            signOut();
            break;
        // ...
    }
}
```

```
private void signOut() {
    mGoogleSignInClient.signOut()
        .addOnCompleteListener(this, new OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                // ...
            }
        });
}
```

# Desconectar cuenta de usuario

Es altamente recomendable permitir desconectar nuestra aplicación de la cuenta del usuario. Podemos añadir un botón de desconexión que al pulsarlo llamaría al método *revokeAccess* que solicitaría la desconexión de la cuenta del usuario de nuestra aplicación.

```
private void revokeAccess() {  
    mGoogleSignInClient.revokeAccess()  
        .addOnCompleteListener(this, new OnCompleteListener<Void>() {  
            @Override  
            public void onComplete(@NonNull Task<Void> task) {  
                // ...  
            }  
        });  
}
```





¿Preguntas...?