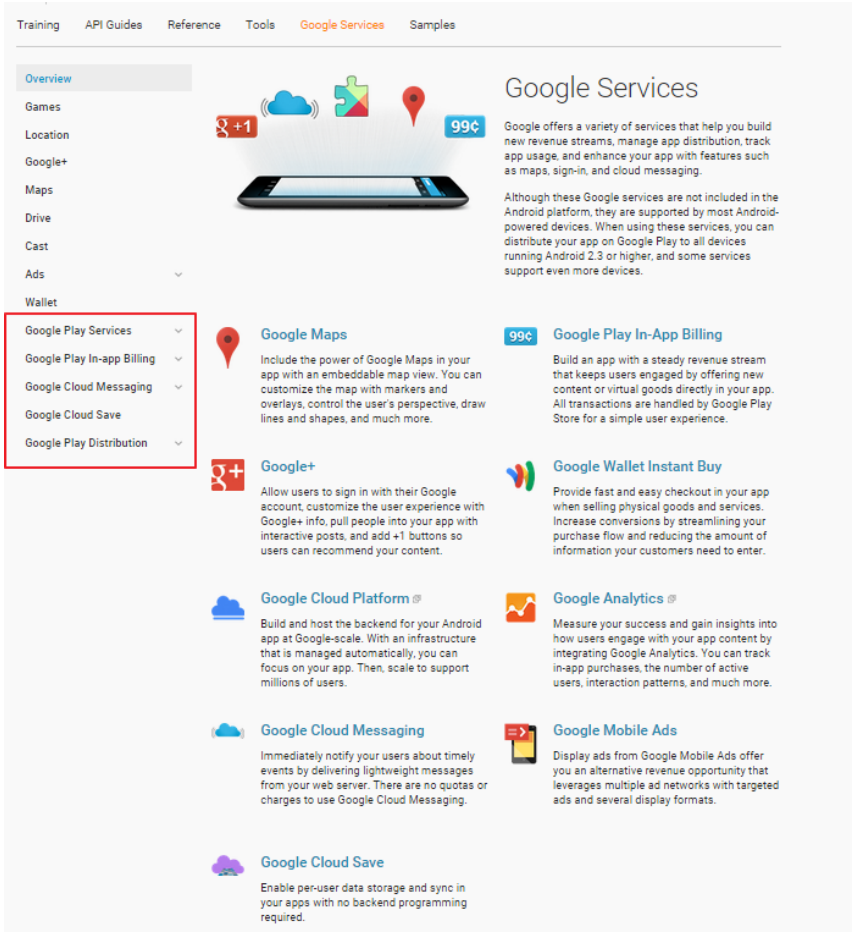
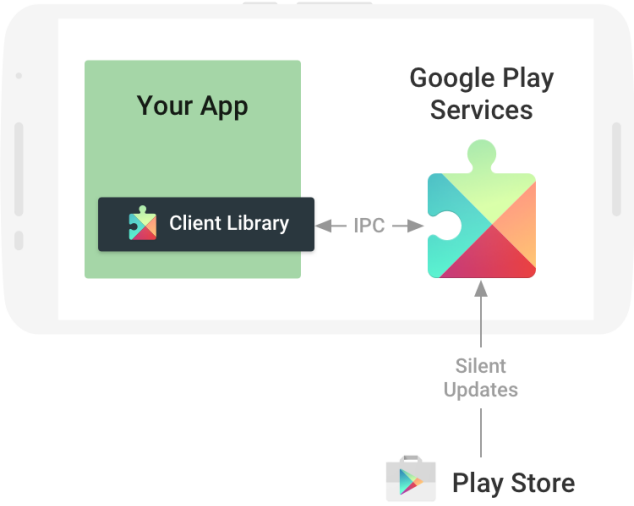
# Google Play Services

# Google Services




# Google Play Services

- Desde hace algún tiempo Google está tendiendo a incorporar muchas de sus APIs para desarrolladores Android (la de mapas, la de localización, la de integración con Google+, etc.) dentro de los llamados *Google Play Services*.
- Los servicios de Google Play "*viven*" como una aplicación más en todos los dispositivos Android, lo que nos aporta la ventaja de no tener que preocuparnos de ellos, ya que son actualizados automáticamente por la plataforma cuando existen novedades.
- Por decirlo de alguna forma, nosotros tan sólo nos tendremos que preocupar de "conectarnos" a dichos servicios que se ejecutan en segundo plano y utilizar las distintas funcionalidades como si fueran parte de nuestra propia aplicación.

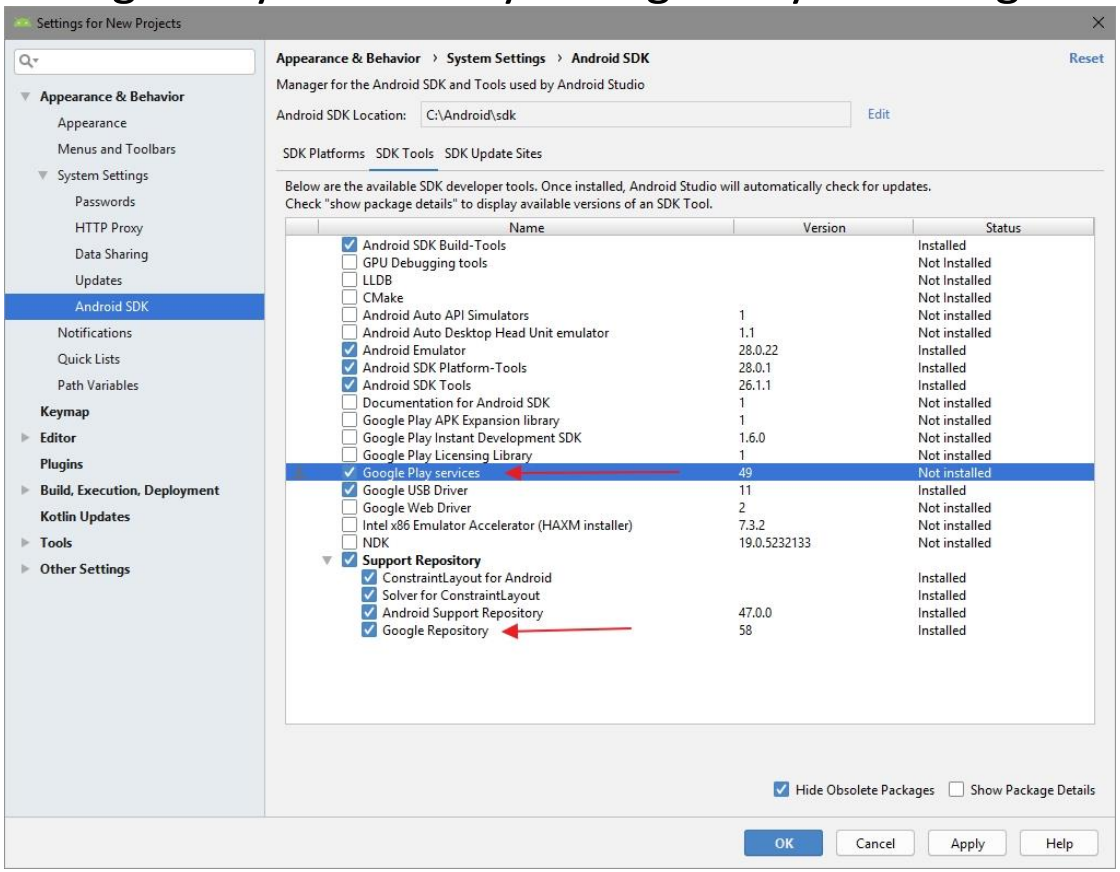


# Preparación de proyectos que usan API's de Google para Android

- Para utilizar las API's de Google es necesario:
  1. Instalar API's de Google Play Services desde SDK Manager.
  2. Probar en dispositivos que funcionen con Android 4.0+ y que tengan instalado Google Play Store o en emuladores Android 4.2.2+ con Google API's
  3. Preparar convenientemente el proyecto: añadir librería con las API's, configurar el Manifest, crear excepciones.
  4. Dependiendo del tipo de servicio se requieren pasos adicionales, por ejemplo el uso de Google Maps requerirá la obtención de una clave de acceso que nos permita el uso de su API. (Android Studio y sus asistentes nos facilitan algunas de estas tareas ☺!)

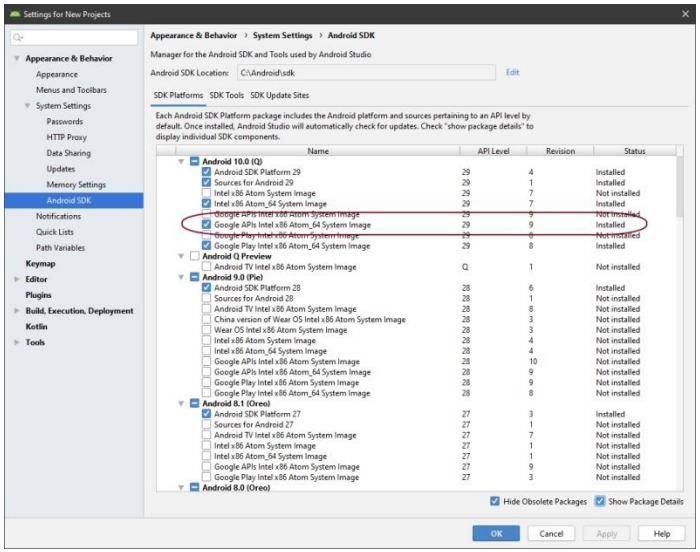
# Conseguir el paquete Google Play Services

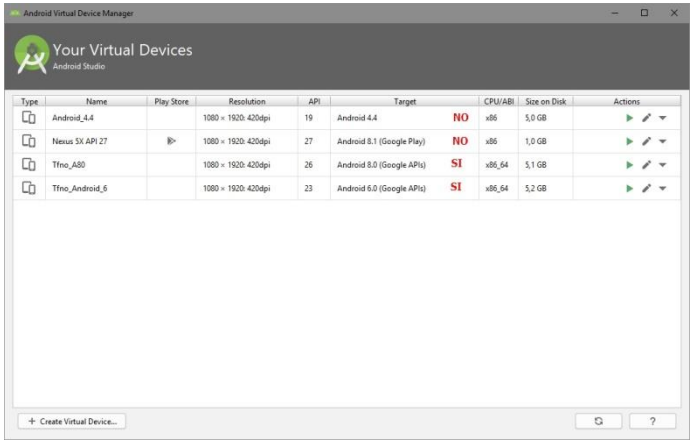
- Accedemos al Android SDK Manager y descargamos desde SDK Tools los paquetes "Google Play Services" y "Google Play Licensing Library".



# Emuladores del SDK válidos

- Las Google API's no son código abierto pero Google ofrece una imagen del sistema con ellas.
- Son válidos los emuladores con un AVD que ejecuta la plataforma de Google APIs basado en Android 4.2.2 o superior. Para conseguirlos desde el SDK hay que instalar

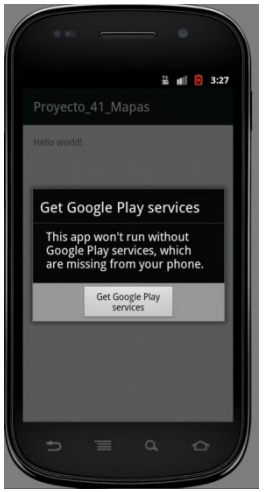


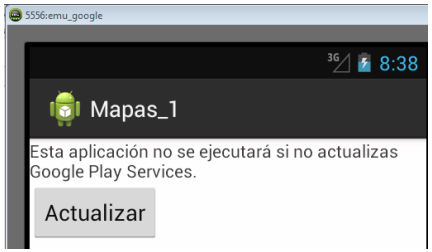



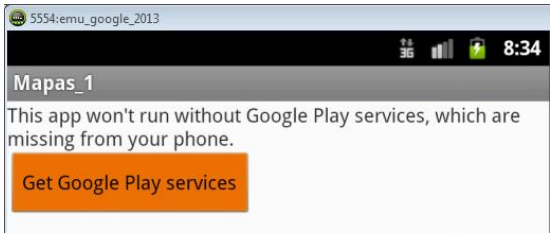

- Los emuladores que utilicemos deberán trabajar con dichos paquetes.

# Problemas con el emulador de Android SDK

- En algunos dispositivos (reales o emulados) al intentar lanzar la aplicación, solicita la actualización de Google Play Services para que pueda ejecutarse.
- Si es un dispositivo real bastará con actualizar, pero en algunos AVD no está instalado Google Play.
- Hay muchos manuales de cómo conseguir instalar Google Play en emuladores instalando las apk necesarias vía adb.



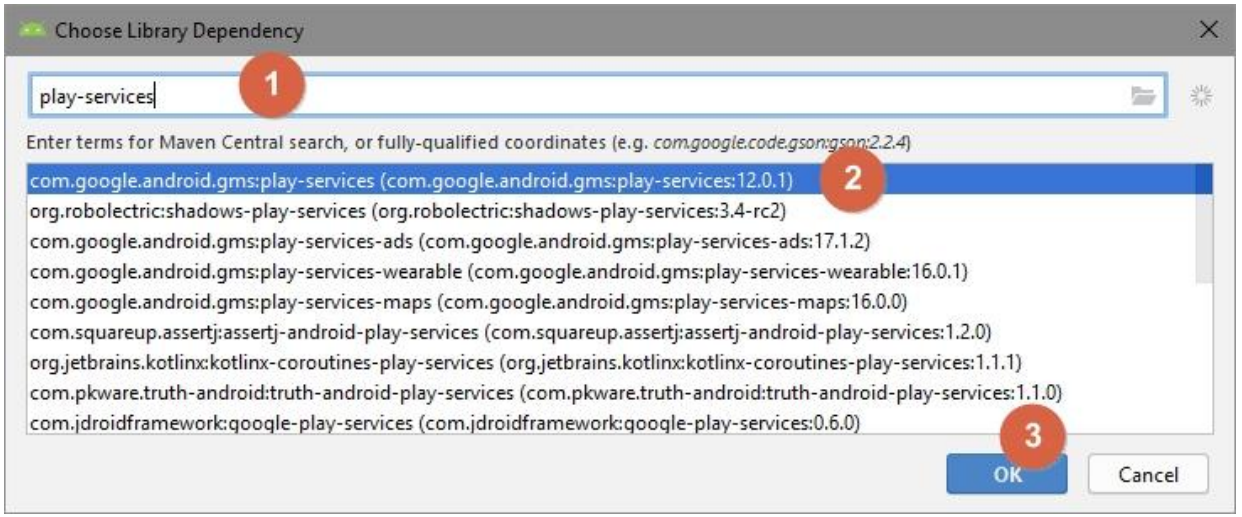



# Usando GenyMotion

- Si usas GenyMotion, tienes que instalar Google Services:
  - tutorial1
  - tutorial2
  - Y otro más

# Añadir las librerías al proyecto en Android Studio

- En Android Studio:




- Puedes comprobar que en el fichero buid.gradle de la app se han añadido las dependencias

```
dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.android.support:design:28.0.0'
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
    implementation 'com.google.android.gms:play-services:12.0.1'
}
```

# Añadir SOLO las librería necesarias

| API | Description in build.gradle |
|---|---|
| Google+ | `com.google.android.gms:play-services-plus:17.0.0` |
| Google Account Login | `com.google.android.gms:play-services-auth:17.0.0` |
| Google Actions, Base Client Library | `com.google.android.gms:play-services-base:17.1.0` |
| Google Sign In | `com.google.android.gms:play-services-identity:17.0.0` |
| Google Analytics | `com.google.android.gms:play-services-analytics:17.0.0` |
| Google Awareness | `com.google.android.gms:play-services-awareness:17.1.0` |
| Google Cast | `com.google.android.gms:play-services-cast:18.0.0` |
| Google Cloud Messaging | `com.google.android.gms:play-services-gcm:17.0.0` |
| Google Drive | `com.google.android.gms:play-services-drive:17.0.0` |
| Google Fit | `com.google.android.gms:play-services-fitness:18.0.0` |
| Google Location and Activity Recognition | `com.google.android.gms:play-services-location:17.0.0` |
| Google Mobile Ads | `com.google.android.gms:play-services-ads:18.3.0` |
| Mobile Vision | `com.google.android.gms:play-services-vision:19.0.0` |
| Google Nearby | `com.google.android.gms:play-services-nearby:17.0.0` |
| Google Panorama Viewer | `com.google.android.gms:play-services-panorama:17.0.0` |
| Google Play Game services | `com.google.android.gms:play-services-games:19.0.0` |
| SafetyNet | `com.google.android.gms:play-services-safetynet:17.0.0` |
| Google Pay | `com.google.android.gms:play-services-wallet:18.0.0` |
| Wear OS by Google | `com.google.android.gms:play-services-wearable:17.0.0` |

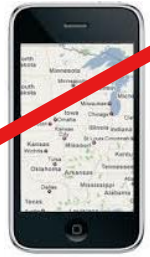
- Si el número de referencias a métodos en una aplicación Android supera el límite de 65K, puede fallar al compilar.
- En vez de incluir toda la librería, se debe incluir solo las que se refieran a los servicios necesarios en la aplicación.

**Selectively compiling APIs into your executable**

Table 1 shows a list of the separate APIs that you can include when compiling your app, and how to describe them in your `build.gradle` file.

★ **Note:** Don't use the combined `play-services` target. It brings in dozens of libraries, bloating your application. Instead, specify only the specific Google Play services APIs your app uses.

# Desajustes de librerías



```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation 'com.android.support:appcompat-v7:28.0.0'
```

All com.android.support libraries must use the exact same version specification (mixing versions can lead to runtime crashes). Found versions 28.0.0, 26.1.0. Examples include com.android.support:animated-vector-drawable:28.0.0 and com.android.support:support-media-compat:26.1.0 less... (Ctrl+F1)
Inspection info:There are some combinations of libraries, or tools and libraries, that are incompatible, or can lead to bugs. One such incompatibility is compiling with a version of the Android support libraries that is not the latest version (or in particular, a version lower than your targetSdkVersion).

Issue id: GradleCompatible

- Solución: añadir

  implementation **'com.android.support:support-v4:28.0.0'**
- Si el número de referencias a métodos en una aplicación Android supera el límite de 65K, puede fallar al compilar.
- En vez de incluir toda la librería, se debe incluir solo las que se refieran a los servicios necesarios en la aplicación.

# Asegurar que el dispositivo está preparado

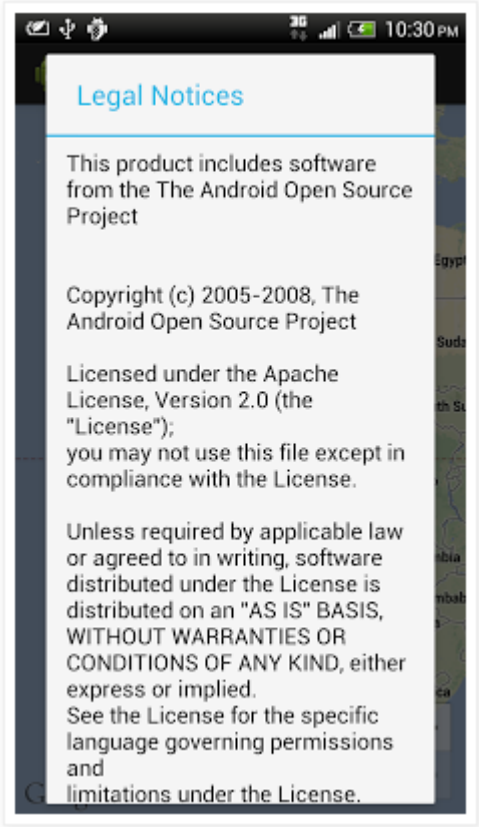
- Google Play ofrece actualizaciones de sus servicios a través de la Google Play Store.
- Debido a que es difícil anticipar el estado de cada dispositivo, *es conveniente* consultar si es compatible con la versión de la API de la aplicación antes de acceder a ellas.

```
boolean valido=comprobarValidez();
    if (valido){
       // seguir con las acciones
    }
    else
       finish();
 }

private boolean comprobarValidez() {
    GoogleApiAvailability googleApiAvailability = GoogleApiAvailability.getInstance();
    int resultCode = googleApiAvailability.isGooglePlayServicesAvailable(this);
    if (resultCode != ConnectionResult.SUCCESS) {
       if (resultCode == ConnectionResult.SERVICE_INVALID) {
          Toast.makeText(this, "Dispositivo sin Google Play Services", Toast.LENGTH_LONG).show();
          return false;
       }
       if (googleApiAvailability.isUserResolvableError(resultCode)) {
          googleApiAvailability.getErrorDialog(this, resultCode, 1000).show();
       } else {
          Toast.makeText(this, "Otros problemas ", Toast.LENGTH_LONG).show();
          return false;
       }
    }
    return true;
 }
```

Esto devuelve un diálogo que proporciona un mensaje adecuado acerca del error y ofrece una acción que lleva al usuario a Google Play Store para instalar la actualización o a la configuración para habilitarlo.
En el ejemplo, 1000 es el código de requestCode. Más información

# Publicar en Google Play




- Si utilizamos las APIs de Google Play Services en aplicaciones que vamos a subir a "Google Play" es obligatorio/recomendado incluir una sección con un "Aviso legal".
- El texto puede obtenerse llamando al método GoogleApiAvailability.getOpenSourceSoftwareLicenseInfo
- Incluir en menú un item llamado menu_legalnotices (por ej.):

```
case R.id.menu_legalnotices:
    String Licencia=
GoogleApiAvailability.getOpenSourceSoftwareLicenseInfo(
     getApplicationContext());
    AlertDialog.Builder LicenseDialog = new
AlertDialog.Builder(MainActivity.this);
    LicenseDialog.setTitle("Legal Notices");
    LicenseDialog.setMessage(Licencia);
    LicenseDialog.show();
    return true;
```

# Accediendo a los servicios de Google - I

- Para acceder a un servicio que no requiere autorización API, se obtiene una instancia del objeto cliente del servicio, pasando el Context o la Activity.

Por ejemplo, para obtener la última ubicación conocida del dispositivo utilizando el servicio de ubicación

```
FusedLocationClient client = LocationServices.getFusedLocationProviderClient(this);

// Get the last known location
client.getLastLocation()
        .addOnCompleteListener(this, new OnCompleteListener<Location>() {
            @Override
            public void onComplete(@NonNull Task<Location> task) {
                // ...
            }
        });
```

# Accediendo a los servicios de Google - II

## Accessing Google services that require authorization

To access a service that requires user authorization, first sign the user in, and request permission to access the scopes required by the service. Then, get an instance of the service's client object, passing it the user's `GoogleSignInAccount` object in addition to a `Context` or `Activity`.

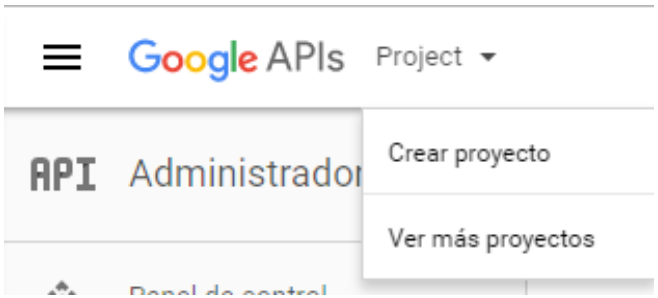
For example, to access app files in a user's Drive:

```
// This account must have the necessary scopes to make the API call
// See https://developers.google.com/identity/sign-in/android/
GoogleSignInAccount account = GoogleSignIn.getLastSignedInAccount(this);

// Get the app's Drive folder
DriveResourceClient client = Drive.getDriveResourceClient(this, account);
client.getAppFolder().
        .addOnCompleteListener(this, new OnCompleteListener<DriveFolder>() {
            @Override
            public void onComplete(@NonNull Task<DriveFolder>() {
                // ...
            }
        });
```

# API Key

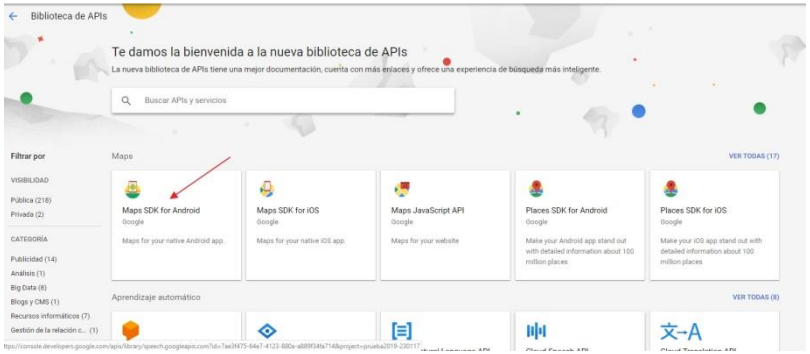
- API Key ("application programing interface key") es un código generado por sitios web para permitir a los usuarios acceder a su aplicación.
- Para poder usar muchos de los servicios de Google Play es necesario obtener la API Key de la aplicación que estará asociada al certificado con el que firmamos digitalmente nuestra aplicación. Esto quiere decir que si cambiamos el certificado con el que firmamos nuestra aplicación (algo que se hace normalmente como paso previo a la publicación de la aplicación en el *market* Google Play) tendremos que cambiar también la clave de uso de la API.
- La obtención de la API Key para servicios de Google Play se ha integrado en la Consola de APIs de Google, por lo que el primer paso será acceder a ella.

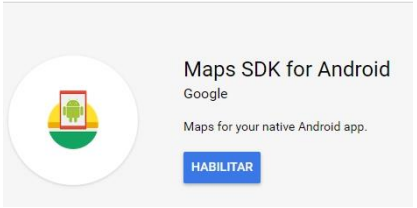



- Una vez hemos accedido, tendremos que crear un nuevo proyecto desplegando el menú superior izquierdo y seleccionando la opción “Create…” y en la siguiente ventana dar un nombre al proyecto.

# Fijar servicios Google

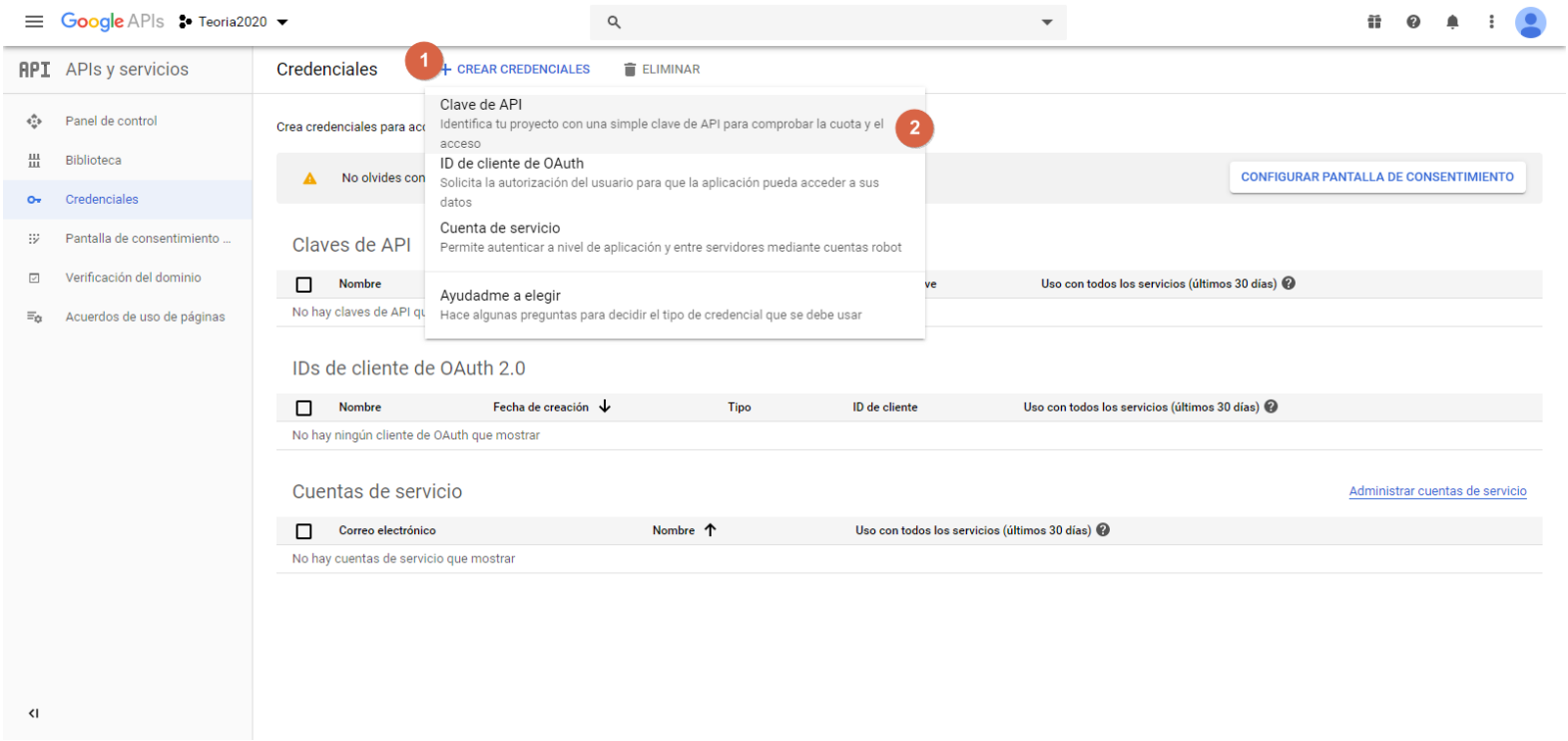
- Una vez creado el proyecto, desde el icono de menú accederemos a APIs y servicios.
- Desde esta ventana podemos activar o desactivar cada uno de los servicios de Google que queremos utilizar.
- Por ejemplo, activaremos el servicio llamado "Maps SDK for Android".



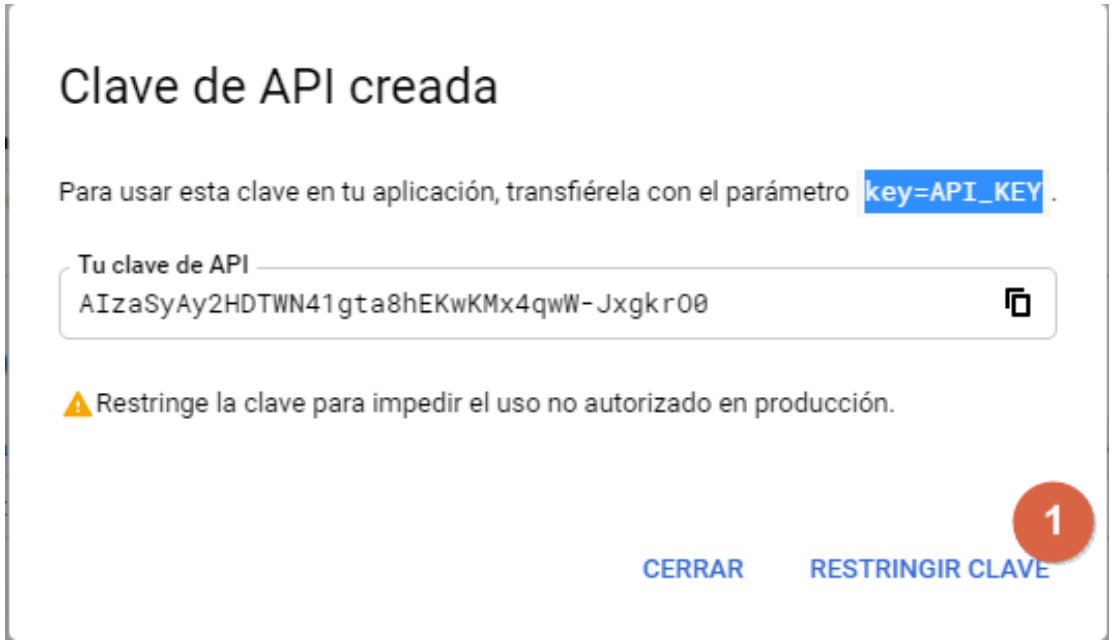
# API Access

- Una vez activado accedemos a una nueva opción en el menú de APIS y Servicios llamada “Credentials” que nos da la posibilidad de obtener nuestra API Key que nos permita utilizar el servicio de mapas desde nuestra aplicación particular.



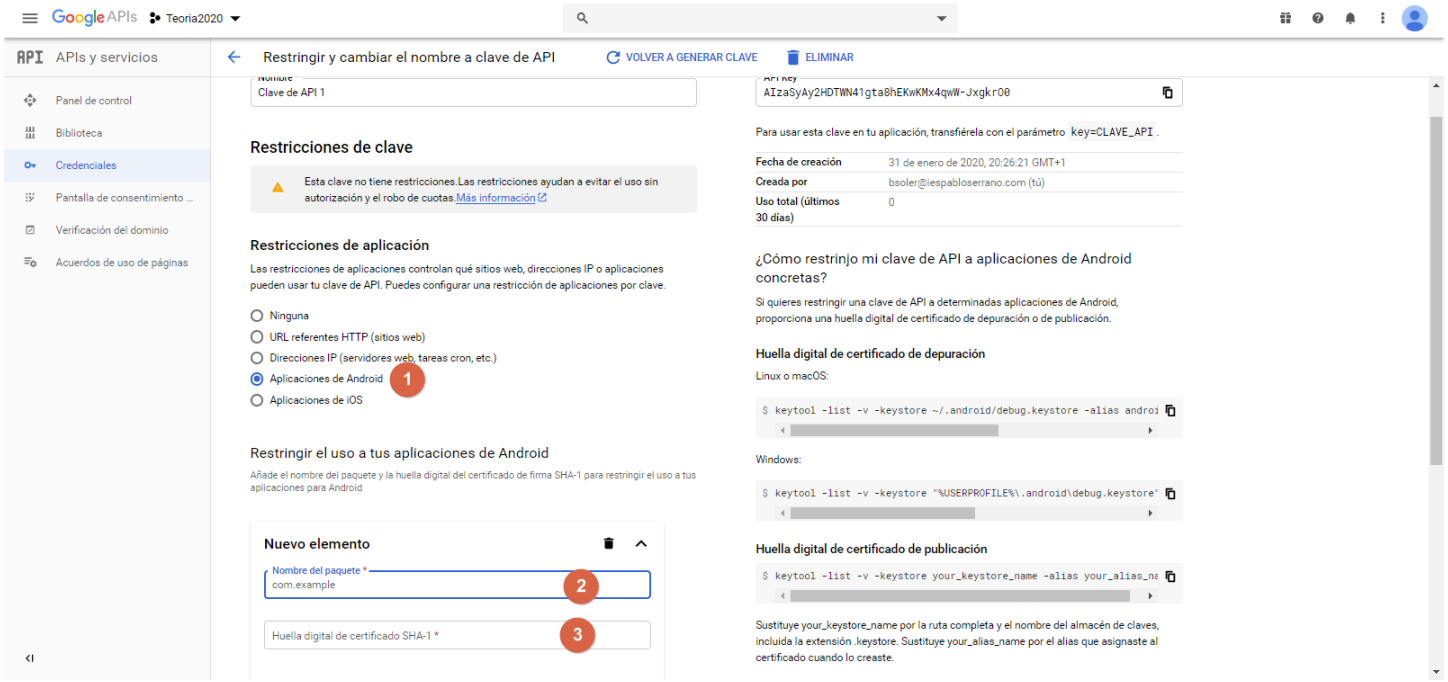
# Usar en aplicación Android

- Tras obtener la "Clave de API" la **restringiremos para Android**.



# Datos de la aplicación

- Tendremos que introducir los siguientes datos identificativos de nuestra aplicación:
  - la huella digital (SHA1) del certificado con el que firmamos la aplicación
  - el nombre del paquete de nuestra aplicación.



# Huella digital del certificado

- Toda aplicación Android debe ir firmada para poder ejecutarse en un dispositivo, tanto físico como emulado.
- Existen dos clases de firmas:
  - Durante el desarrollo, para realizar pruebas y la depuración del código, aunque no seamos conscientes de ello también estamos firmado la aplicación con un "certificado de pruebas". Solo será válida para instalar vía USB.
  - La de publicación o "certificado de *release*". Se usa para instalar vía Gooogle Play.

# Obtener la huella digital del certificado

- IMPORTANTE: Hay que usar keytool que está en la carpeta

  Android Studio/jre/bin

  No el de la carpeta c:/java/jdk/bin o c:/java/jre/bin
- Para obtener el certificado de desarrollo: desde la consola de Windows (cmd):
  - keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypass android

```
Huellas digitales del Certificado:
         MD5: 38:BA:60:E0:12:BD:AB:49:C0:BA:61:30:29:0C:1E:E5
         SHA1: CF:9D:59:52:0A:ED:F4:6E:AF:17:1D:C7:F2:5D:18:66:9C:3E:05:32
         SHA256: 32:5D:F6:72:DC:C8:34:EE:C8:94:4F:20:0D:37:04:3B:39:1E:EE:42:4F
```

- Para obtener el certificado de desarrollo: Linux
  - keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android

# Identificando la aplicación

- Una vez introducida nos devuelve la clave para nuestra aplicación:

| | |
|---|---|
| **Clave de API** | AIzaSyD9J1IDWCsvsHbCOcib2uShmSI5NZR |

- Consejo: No crees muchos proyectos (gratis son un número limitado), dentro de un mismo proyecto puedes tener varias aplicaciones distintas!

# Añadir la API Key al AndroidManifest.xml

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
<!--
        The API key for Google Maps-based APIs is defined as a string resource.
        (See the file "res/values/google_maps_api.xml").
        Note that the API key is linked to the encryption key used to sign the APK.
        You need a different API key for each encryption key, including the release key that is used to
        sign the APK for publishing.
        You can define the keys for the debug and release targets in src/debug/ and src/release/.
    -->
    <meta-data
      android:name="com.google.android.geo.API_KEY"
      android:value="@string/google_maps_key"/>

    <activity
     …
```