

# Mapas

Profesor: Ana Isabel Vegas

# INDICE

<b>1. INTRODUCCIÓN .....</b>	<b>3</b>
<b>DESCARGAR GOOGLE API.....</b>	<b>3</b>
<b>OBTENER MAPKEY .....</b>	<b>4</b>
<b>2.- MOSTRAR EL MAPA .....</b>	<b>9</b>
<b>3.- OPERACIONES SOBRE EL MAPA .....</b>	<b>12</b>
<b>CAMBIAR DE VISTA .....</b>	<b>12</b>
<b>MOVIMIENTO DE LA CAMARA .....</b>	<b>13</b>
<b>OBTENER LA LOCALIZACION .....</b>	<b>14</b>
<b>ÍNDICE DE GRÁFICOS .....</b>	<b>16</b>

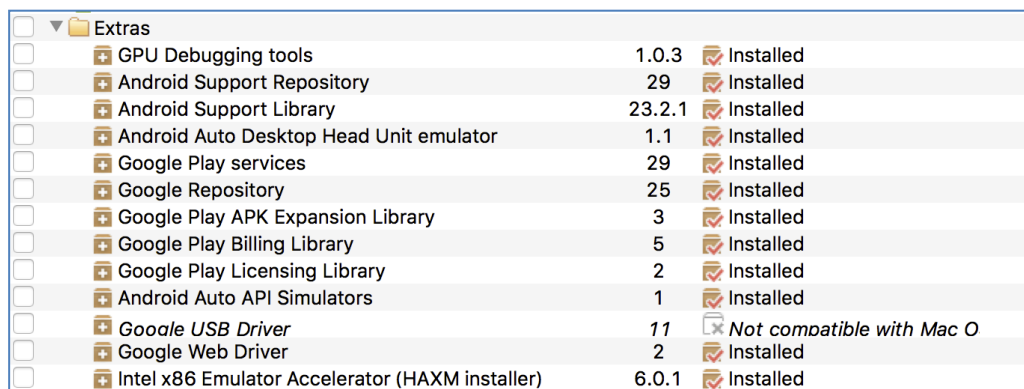
# 1. INTRODUCCIÓN

Gracias a la nueva versión de la API de mapas de Google (Google Maps Android API v2) podemos visualizar mapas en nuestro dispositivo.

Necesitamos previamente hacer una serie de pasos para configurar el entorno y que todo funcione correctamente.

## DESCARGAR GOOGLE API

En primer lugar, dado que la API v2 se proporciona como parte del SDK de Google Play Services, será necesario incorporar previamente a nuestro entorno de desarrollo dicho paquete. Haremos esto accediendo desde Eclipse al Android SDK Manager y descargando del apartado de extras el paquete llamado “Google Play Services”.



<input type="checkbox"/>	Extras		
<input type="checkbox"/>	GPU Debugging tools	1.0.3	Installed
<input type="checkbox"/>	Android Support Repository	29	Installed
<input type="checkbox"/>	Android Support Library	23.2.1	Installed
<input type="checkbox"/>	Android Auto Desktop Head Unit emulator	1.1	Installed
<input type="checkbox"/>	Google Play services	29	Installed
<input type="checkbox"/>	Google Repository	25	Installed
<input type="checkbox"/>	Google Play APK Expansion Library	3	Installed
<input type="checkbox"/>	Google Play Billing Library	5	Installed
<input type="checkbox"/>	Google Play Licensing Library	2	Installed
<input type="checkbox"/>	Android Auto API Simulators	1	Installed
<input type="checkbox"/>	Google USB Driver	11	Not compatible with Mac OS
<input type="checkbox"/>	Google Web Driver	2	Installed
<input type="checkbox"/>	Intel x86 Emulator Accelerator (HAXM installer)	6.0.1	Installed

Gráfico 1. Paquete “Google Play Services”.

## OBTENER MAPKEY

Si creamos un proyecto para utilizar mapas se generará el archivo `google_maps_api.xml` donde debemos introducir la clave que obtenemos a través de la consola de google.

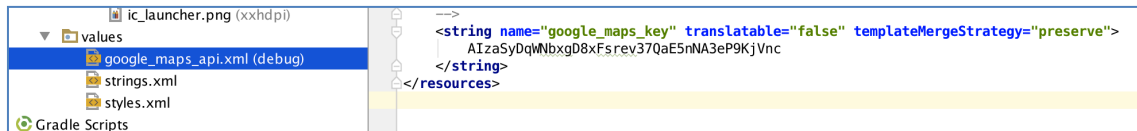


Gráfico 2. Archivo `google_maps_api.xml`

El siguiente paso será obtener una *API Key* para poder utilizar el servicio de mapas de Google en nuestra aplicación.

Toda aplicación Android debe ir firmada para poder ejecutarse en un dispositivo, tanto físico como emulador. Este proceso de firma es uno de los pasos que tenemos que hacer siempre antes de distribuir públicamente una aplicación.

Adicionalmente, durante el desarrollo de la misma, para realizar pruebas y la depuración del código, aunque no seamos conscientes de ello también estamos firmado la aplicación con un “certificado de pruebas”.

Accedemos a la Consola de APIs de Google a través del link que nos indica el xml anterior:

<https://code.google.com/apis/console/?pli=1>

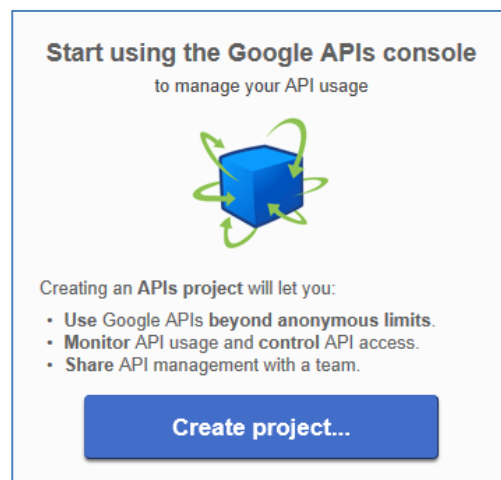


Gráfico 3. Consola API de Google

En la parte izquierda de la ventana hay un desplegable como API Project. Despliego y elijo Create ...

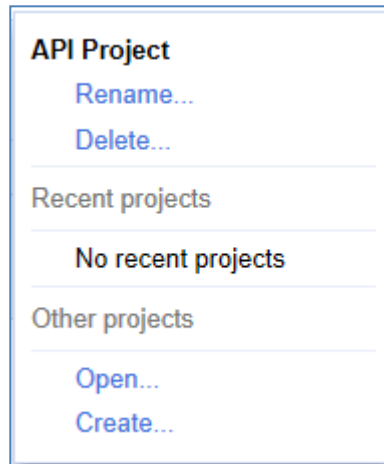


Gráfico 4. Crear un proyecto

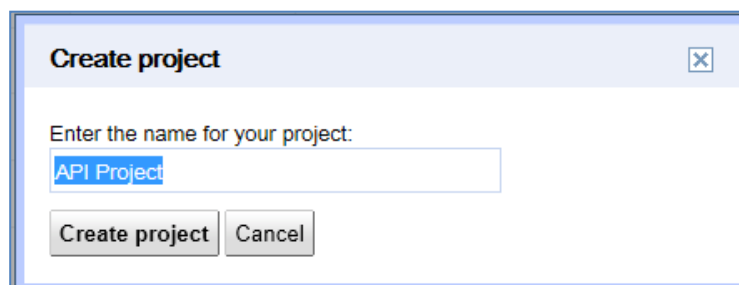


Gráfico 5. Elegimos nombre del proyecto

Una vez creado el proyecto, accederemos a la opción “Services” del menú izquierdo. Desde esta ventana podemos activar o desactivar cada uno de los servicios de Google que queremos utilizar. En este caso sólo activaremos el servicio llamado “Google Maps Android API v2” pulsando sobre el botón ON/OFF situado justo a su derecha.

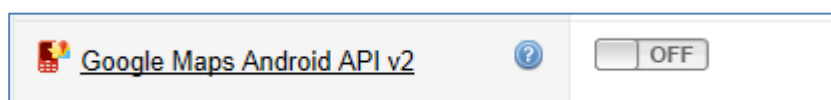


Gráfico 6. Activaremos el servicio llamado “Google Maps Android API v2”

**Google Maps/Google Earth APIs Terms of Service**

Last Updated: May 10, 2013

**1. Your relationship with Google.**

1.1 Use of the Service is Subject to these Terms. Your use of any of the Google Maps/Google Earth APIs (referred to in this document as the **"Maps API(s)"** or the **"Service"**) is subject to the terms of a legal agreement between you and Google (the **"Terms"**). "Google" means either (a) Google Ireland Limited, with offices at Gordon House, Barrow Street, Dublin 4, Ireland, if Customer's billing address is in any country within Europe, the Middle East, or Africa (**"EMEA"**); (b) Google Asia Pacific Pte. Ltd., with offices at 8 Marina View Asia Square 1 #30-01 Singapore 018960, if Customer's billing address is in any country within the Asia Pacific region (**"APAC"**); or (c) Google Inc., with offices at 1600 Amphitheatre Parkway, Mountain View, California 94043, USA, if Customer's billing address is in any country in the world other than those in EMEA and APAC.

1.2 The Terms include Google's Legal Notices and Privacy Policy.

(a) Unless otherwise agreed in writing with Google, the Terms will include the following:

- (i) the terms and conditions set forth in this document (the **"Maps APIs Terms"**);
- (ii) the [Legal Notices](#); and
- (iii) the [Privacy Policy](#).

☒ I agree to these terms.

Gráfico 7. Aceptamos los términos

Una vez activado aparecerá una nueva opción en el menú izquierdo llamada "API Access". Accediendo a dicha opción tendremos la posibilidad de obtener nuestra nueva API Key que nos permita utilizar el servicio de mapas desde nuestra aplicación particular.

**API Project**

- Overview
- Services
- Team
- API Access**
- Reports
- Quotas

**API Access**

To prevent abuse, Google places limits on API requests. Using a valid OAuth token or API key allows you to exceed anonymous limits by connecting requests back to your project.

**Authorized API Access**

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private. A single project may contain up to 20 client IDs. [Learn more](#)

**Simple API Access**

Use API keys to identify your project when you do not need to access user data. [Learn more](#)

**Key for browser apps (with referers)**

API key: `AIzaSyCg0JvNdy_TFn6V0lx18ubE6pvcSNawkeU`

Referers: Any referer allowed

Activated on: May 20, 2013 10:33 AM

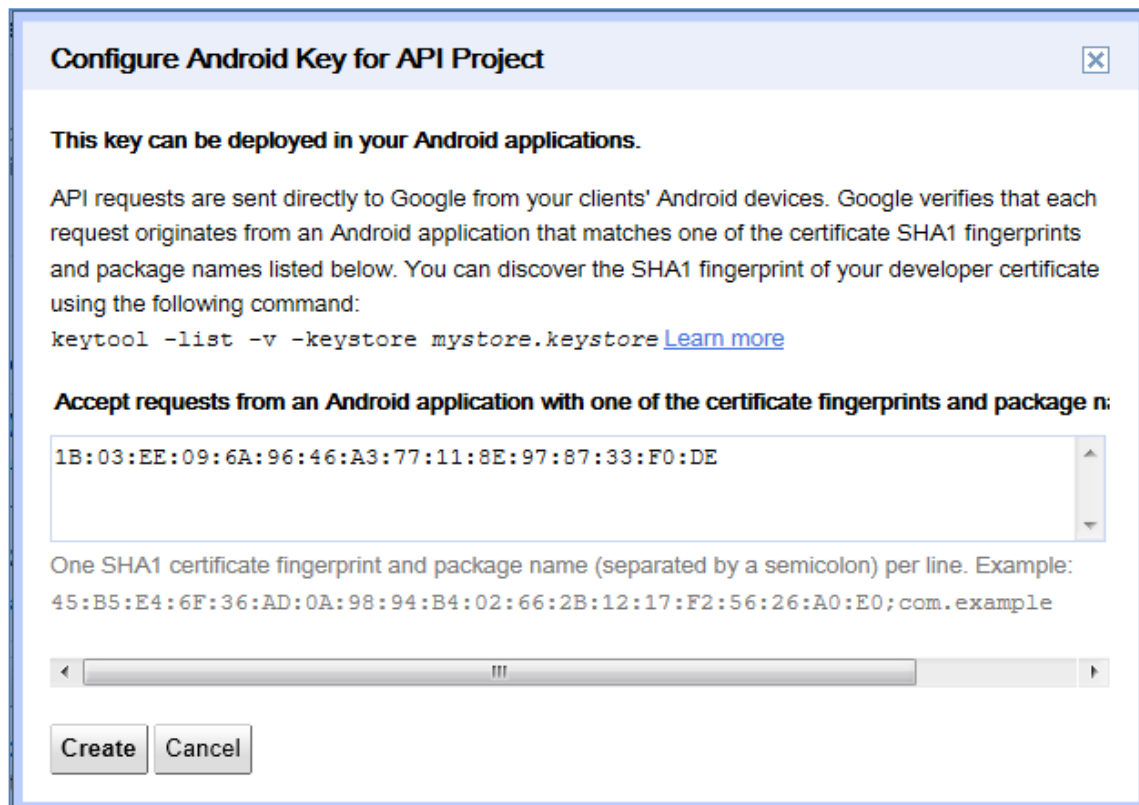
Activated by: `anaisabelveg@gmail.com` - **you**

Gráfico 8. Ventana para obtener la clave

Pulsaremos el botón "Create new Android key...". Esto nos llevará a un cuadro de diálogo donde tendremos que introducir algunos datos identificativos de nuestra aplicación.

En concreto necesitaremos dos: la huella digital (SHA1) del certificado con el que firmamos la aplicación anteriormente, y el paquete java utilizado en la aplicación que vamos a desarrollar.

Aunque en la imagen siguiente no lo muestra después de la clave separado por punto y coma (;) iría el paquete de la aplicación.



**Configure Android Key for API Project**

This key can be deployed in your Android applications.

API requests are sent directly to Google from your clients' Android devices. Google verifies that each request originates from an Android application that matches one of the certificate SHA1 fingerprints and package names listed below. You can discover the SHA1 fingerprint of your developer certificate using the following command:

```
keytool -list -v -keystore mystore.keystore Learn more
```

**Accept requests from an Android application with one of the certificate fingerprints and package n:**

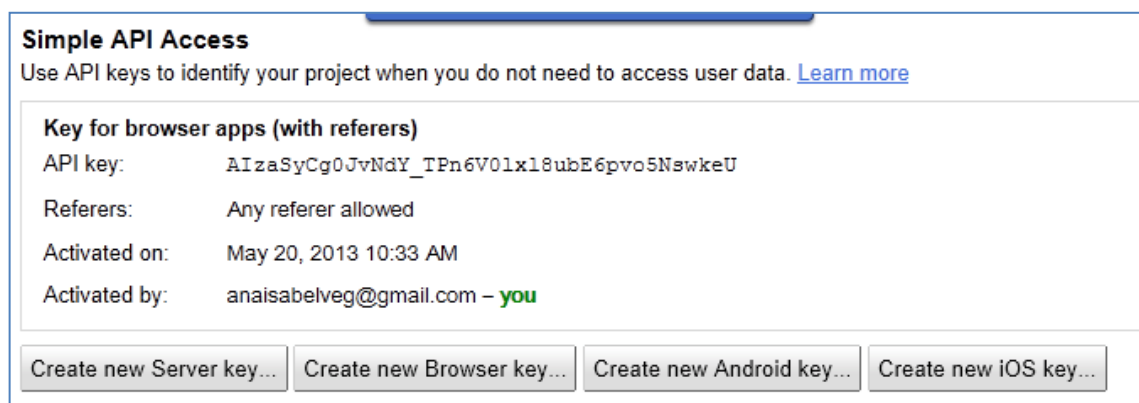
1B:03:EE:09:6A:96:46:A3:77:11:8E:97:87:33:F0:DE

One SHA1 certificate fingerprint and package name (separated by a semicolon) per line. Example:  
45:B5:E4:6F:36:AD:0A:98:94:B4:02:66:2B:12:17:F2:56:26:A0:E0;com.example

Create Cancel

Gráfico 9. Introducimos los datos de nuestra aplicación

Pulsamos el botón “Create” y ya deberíamos tener nuestra API Key generada, podremos verla en la pantalla siguiente dentro del apartado “Key for Android Apps (with certificates)”.



**Simple API Access**

Use API keys to identify your project when you do not need to access user data. [Learn more](#)

**Key for browser apps (with referers)**

API key: AIzaSyCg0JvNdY\_TPn6V01x18ubE6pvo5NswkeU

Referers: Any referer allowed

Activated on: May 20, 2013 10:33 AM

Activated by: anaisabelveg@gmail.com – **you**

Create new Server key... Create new Browser key... Create new Android key... Create new iOS key...

Gráfico 10. Clave generada

La clave generada es: AlzaSyCg0JvNdY\_TPn6V0Ixl8ubE6pvo5NswkeU



## 2.- MOSTRAR EL MAPA

Una vez que ya hemos obtenido la clave ya podemos generarnos el proyecto. Tras esto lo primero que haremos será añadir al fichero `google_maps_api.xml` la API Key que acabamos de generar.

```
<string name="google_maps_key" translatable="false" templateMergeStrategy="preserve">
    AIzaSyCmfESP01ID2ms_AOX54PhuRG9cMfd5b8M
</string>
```

Gráfico 11. Clave obtenida

También tendremos que incluir una serie de permisos que nos permitan hacer uso de los mapas.

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
<!--
The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
Google Maps Android API v2, but are recommended.
-->
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Gráfico 12. Permisos establecidos

Dado que la API v2 de Google Maps Android utiliza OpenGL ES versión 2, deberemos especificar también dicho requisito en nuestro `AndroidManifest` añadiendo un nuevo elemento `<uses-feature>`:

```
<uses-feature android:glEsVersion="0x00020000" android:required="true" />
```

Gráfico 13. Configuración de OpenGL

Debemos referenciar desde nuestro proyecto la librería con el SDK de Google Play Services en el archivo de gradle.

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.1.1'
    compile 'com.google.android.gms:play-services:7.3.0'
}
```

Gráfico 14. Librería android-support-v4.jar

A continuación nos diseñamos el layout:

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity" />
```

Gráfico 15. Layout para mostrar el mapa.

Por supuesto, dado que estamos utilizando fragments, la actividad principal también tendrá que extender a `FragmentActivity` (en vez de simplemente `Activity` como es lo “normal”). Usaremos también la versión de `FragmentActivity` incluida en la librería `android-support` para ser compatibles con la mayoría de las versiones Android actuales.

```
package com.example.mapas;

import android.os.Bundle;

public class MainActivity extends android.support.v4.app.FragmentActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

Gráfico 16. Activity

Con esto, ya podríamos ejecutar y probar nuestra aplicación.



Gráfico 17. Mapa en el dispositivo

## 3.- OPERACIONES SOBRE EL MAPA

Accederemos a este componente llamando al método `getMap()` del fragmento `MapFragment` que contenga nuestro mapa.

```
private void setUpMapIfNeeded() {  
    // Do a null check to confirm that we have not already instantiated the map.  
    if (mMap == null) {  
        // Try to obtain the map from the SupportMapFragment.  
        mMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map))  
            .getMap();  
        // Check if we were successful in obtaining the map.  
        if (mMap != null) {  
            setUpMap();  
        }  
    }  
}
```

Gráfico 18. Acceso al componente mapa

## CAMBIAR DE VISTA

Para modificar el tipo de mapa mostrado podremos utilizar una llamada a su método `setMapType()`, pasando como parámetro el tipo de mapa:

- `MAP_TYPE_NORMAL`
- `MAP_TYPE_HYBRID`
- `MAP_TYPE_SATELLITE`
- `MAP_TYPE_TERRAIN`

```
private void alternarVista() {  
    vista = (vista + 1) % 4;  
    switch (vista) {  
        case 0:  
            mapa.setMapType(GoogleMap.MAP_TYPE_NORMAL);  
            break;  
        case 1:  
            mapa.setMapType(GoogleMap.MAP_TYPE_HYBRID);  
            break;  
        case 2:  
            mapa.setMapType(GoogleMap.MAP_TYPE_SATELLITE);  
            break;  
        case 3:  
            mapa.setMapType(GoogleMap.MAP_TYPE_TERRAIN);  
            break;  
    }  
}
```

Gráfico 19. Cambio de tipo de mapa

## MOVIMIENTO DE LA CAMARA

El movimiento de la cámara se va a realizar siempre mediante la construcción de un objeto `CameraUpdate` con los parámetros necesarios. Para los movimientos más básicos como la actualización de la latitud y longitud o el nivel de zoom podremos utilizar la clase `CameraUpdateFactory` y sus métodos estáticos que nos facilitará un poco el trabajo.

Así por ejemplo, para cambiar sólo el nivel de zoom podremos utilizar los siguientes métodos para crear nuestro `CameraUpdate`:

- `CameraUpdateFactory.zoomIn()`. Aumenta en 1 el nivel de zoom.
- `CameraUpdateFactory.zoomOut()`. Disminuye en 1 el nivel de zoom.
- `CameraUpdateFactory.zoomTo(nivel_de_zoom)`. Establece el nivel de zoom.

Por su parte, para actualizar sólo la latitud-longitud de la cámara podremos utilizar:

- `CameraUpdateFactory.newLatLng(lat, long)`. Establece la lat-lng expresadas en grados.

Si queremos modificar los dos parámetros anteriores de forma conjunta, también tendremos disponible el método siguiente:

- `CameraUpdateFactory.newLatLngZoom(lat, long, zoom)`. Establece la lat-lng y el zoom.

Para movernos lateralmente por el mapa (panning) podríamos utilizar los métodos de scroll:

- `CameraUpdateFactory.scrollBy(scrollHorizontal, scrollVertical)`. Scroll expresado en píxeles.

Tras construir el objeto `CameraUpdate` con los parámetros de posición tendremos que llamar a los métodos `moveCamera()` o `animateCamera()` de nuestro objeto `GoogleMap`, dependiendo de si queremos que la actualización de la vista se muestre directamente o de forma animada.

Con esto en cuenta, si quisiéramos por ejemplo centrar la vista en España con un zoom de 5 podríamos hacer lo siguiente:

```
CameraUpdate camUpd1 = CameraUpdateFactory.newLatLng(new LatLng(
    40.41, -3.69));
mapa.moveCamera(camUpd1);
```

Gráfico 20. Mover Cámara

Si queremos modificar los demás parámetros de la cámara o varios de ellos simultáneamente tendremos disponible el método más general `CameraUpdateFactory.newCameraPosition()` que recibe como parámetro un objeto de tipo `CameraPosition`.

Este objeto lo construiremos indicando todos los parámetros de la posición de la cámara a través de su método `Builder()` de la siguiente forma:

```
LatLng madrid = new LatLng(40.417325, -3.683081);
// Centramos el mapa en Madrid
CameraPosition camPos = new CameraPosition.Builder().target(madrid)
    .zoom(19) // Establecemos el zoom en 19
    .bearing(45) // Establecemos la orientación con el noreste arriba
    .tilt(70) // Bajamos el punto de vista de la cámara 70 grados
    .build();
CameraUpdate camUpd3 = CameraUpdateFactory
    .newCameraPosition(camPos);
mapa.animateCamera(camUpd3);
```

Gráfico 21. Cambios en la cámara

## OBTENER LA LOCALIZACION

Si el usuario se mueve de forma manual por él, ¿cómo podemos conocer en un momento dado la posición de la cámara?

Pues igual de fácil, mediante el método `getCameraPosition()`, que nos devuelve un objeto `CameraPosition` como el que ya conocíamos. Accediendo a los

distintos métodos y propiedades de este objeto podemos conocer con exactitud la posición de la cámara, la orientación y el nivel de zoom.

```
CameraPosition camPos = mapa.getCameraPosition();  
  
LatLng coordenadas = camPos.target;  
double latitud = coordenadas.latitude;  
double longitud = coordenadas.longitude;  
  
float zoom = camPos.zoom;  
float orientacion = camPos.bearing;  
float angulo = camPos.titl;
```

Gráfico 22. Obtener la localización

## ÍNDICE DE GRÁFICOS

<b>Gráfico 1. Paquete “Google Play Services” .....</b>	<b>3</b>
<b>Gráfico 2. Archivo google_maps_api.xml .....</b>	<b>4</b>
<b>Gráfico 3. Consola API de Google .....</b>	<b>4</b>
<b>Gráfico 4. Crear un proyecto .....</b>	<b>5</b>
<b>Gráfico 5. Elegimos nombre del proyecto.....</b>	<b>5</b>
<b>Gráfico 6. Activaremos el servicio llamado “Google Maps Android API v2” ..</b>	<b>5</b>
<b>Gráfico 7. Aceptamos los términos .....</b>	<b>6</b>
<b>Gráfico 8. Ventana para obtener la clave .....</b>	<b>6</b>
<b>Gráfico 9. Introducimos los datos de nuestra aplicación .....</b>	<b>7</b>
<b>Gráfico 10. Clave generada .....</b>	<b>7</b>
<b>Gráfico 11. Clave obtenida .....</b>	<b>9</b>
<b>Gráfico 12. Permisos establecidos .....</b>	<b>9</b>
<b>Gráfico 13. Configuración de OpenGL .....</b>	<b>9</b>
<b>Gráfico 14. Librería android-support-v4.jar .....</b>	<b>9</b>
<b>Gráfico 15. Layout para mostrar el mapa. ....</b>	<b>10</b>
<b>Gráfico 16. Activity .....</b>	<b>10</b>
<b>Gráfico 17. Mapa en el dispositivo .....</b>	<b>11</b>
<b>Gráfico 18. Acceso al componente mapa.....</b>	<b>12</b>
<b>Gráfico 19. Cambio de tipo de mapa .....</b>	<b>12</b>
<b>Gráfico 20. Mover Cámara.....</b>	<b>14</b>
<b>Gráfico 21. Cambios en la cámara .....</b>	<b>14</b>
<b>Gráfico 22. Obtener la localización .....</b>	<b>15</b>