1 PDM Avanzado GoogleMaps

1.1 Sumario

1.2 Introdución

Neste punto imos aprender como debuxar e controlar un Mapa de GoogleMaps.

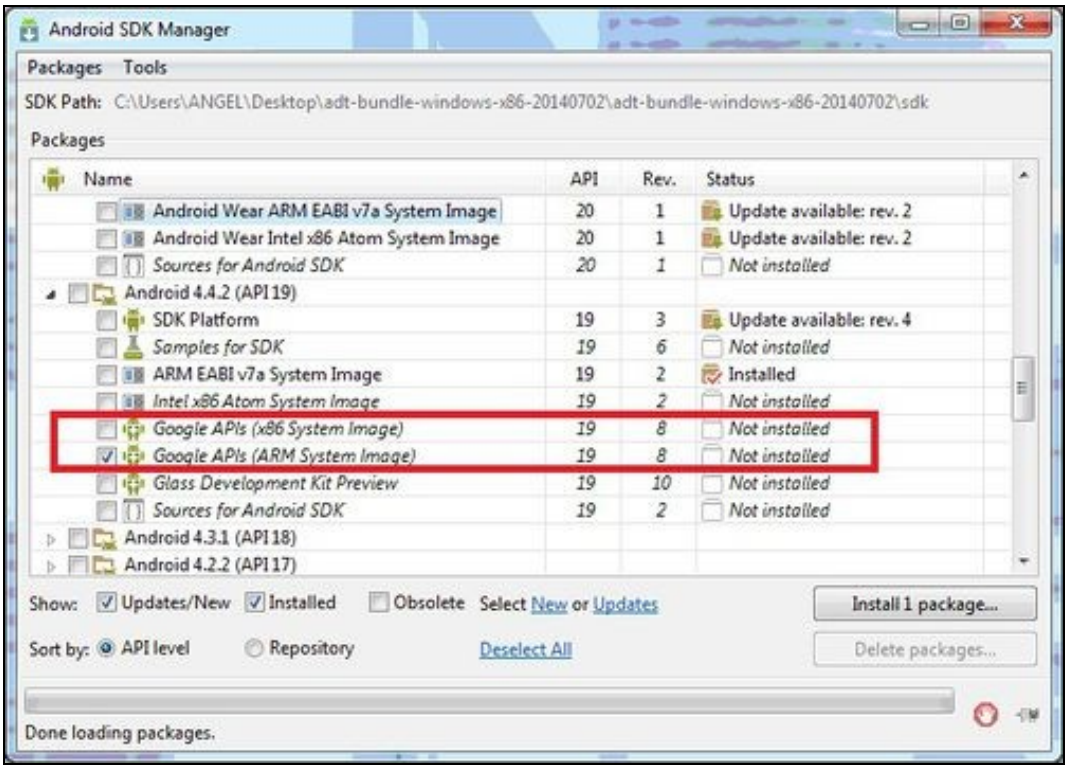
Debido a que o control que debuxa o mapa fai uso dun MapFragment e esta clase foi introducida a partires da API 12 (Android 3.1) teremos dúas formas de utilizalo dependendo da API mínima ó que vaia dirixida a nosa aplicación.

Para poder utilizar un Google Map necesitaremos:

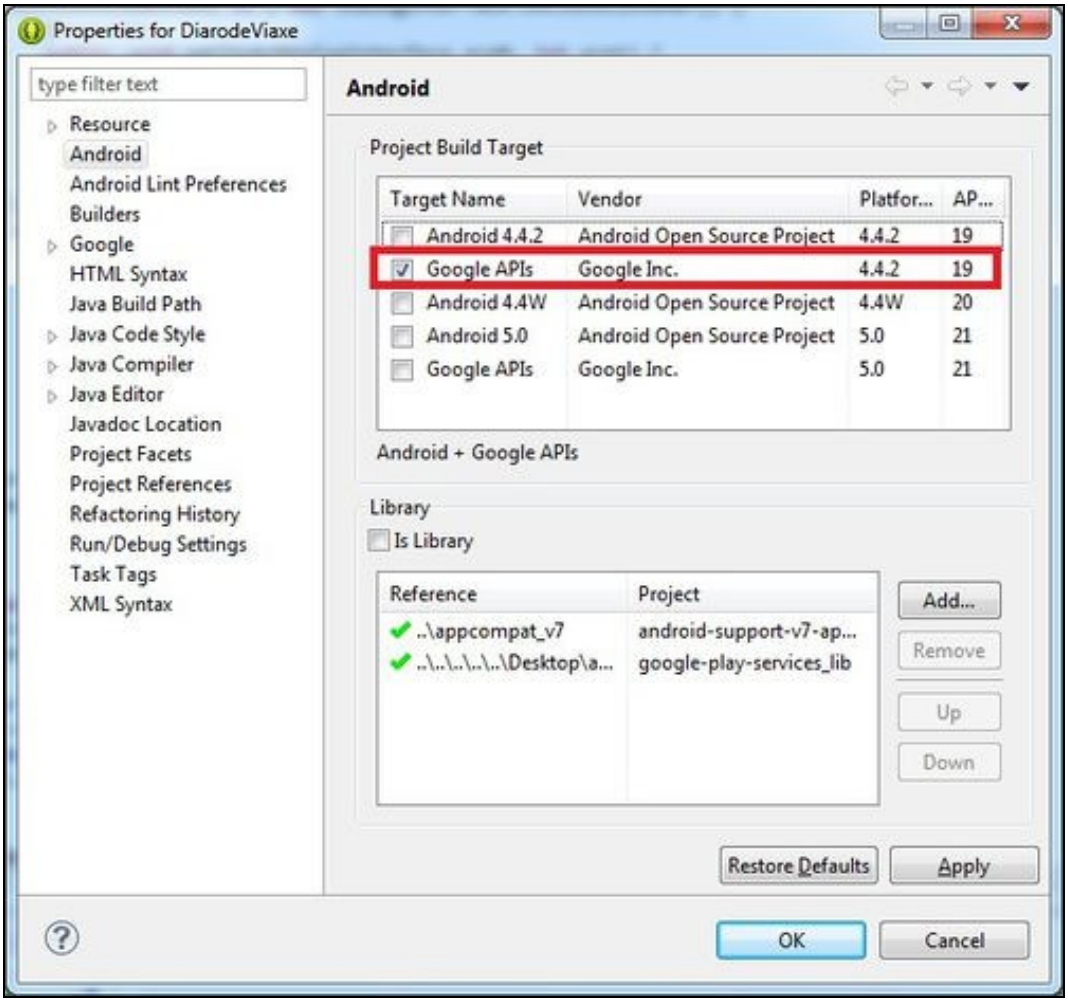
- Google Play services (a aplicación debe estar instalada no dispositivo real. Isto xa se fai automaticamente a partires da versión 2.3).
- Google Maps API key: Unha clave que nos proporciona Google e que ten estar posta no AndroidManifiest.xml. Para ser exactos imos necesitar dúas. Unha para a versión de desenrolo e outra para cando xeremos o APK.
- Unha serie de permisos a engadir no AndroidManifiest.xml

1.2.1 Aclaración sobre o dispositivo virtual android (AVD)

Para poder utilizar o mapa nun AVD é necesario descargar unha imaxe que sexa Google API.



Unha vez descargada e tendo un AVD con esta imaxe, teremos que cambiar as propiedades do proxecto para indicarlle que imos utilizar dita imaxe:



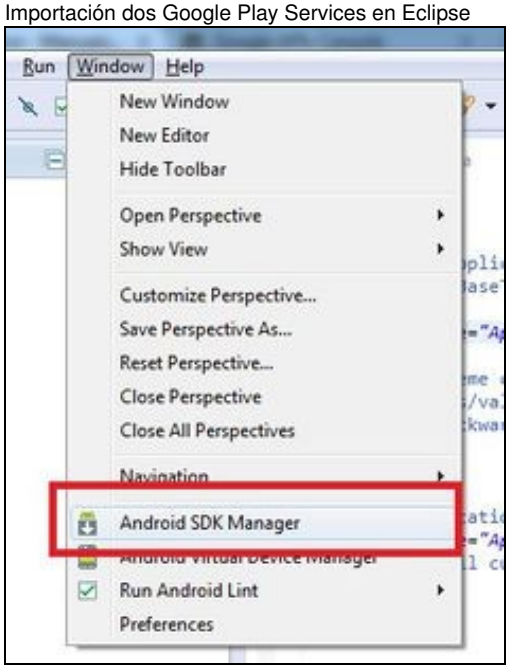
No caso de utilizar a máquina de VirtualBox entregada para este curso, isto non será necesario.

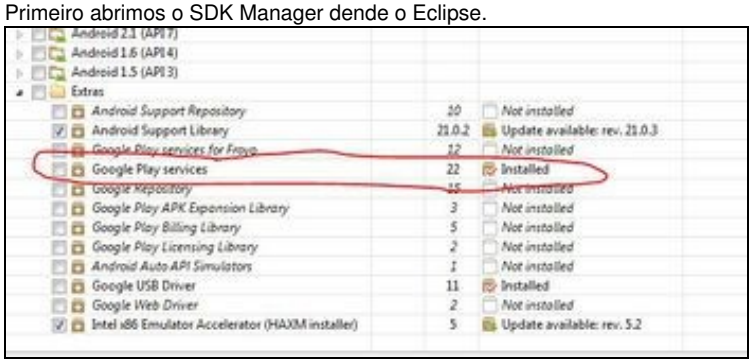
1.3 Importación de Google Play Services ó noso proxecto

Debemos de engadir a librería de Google Play Services ó noso proxecto.

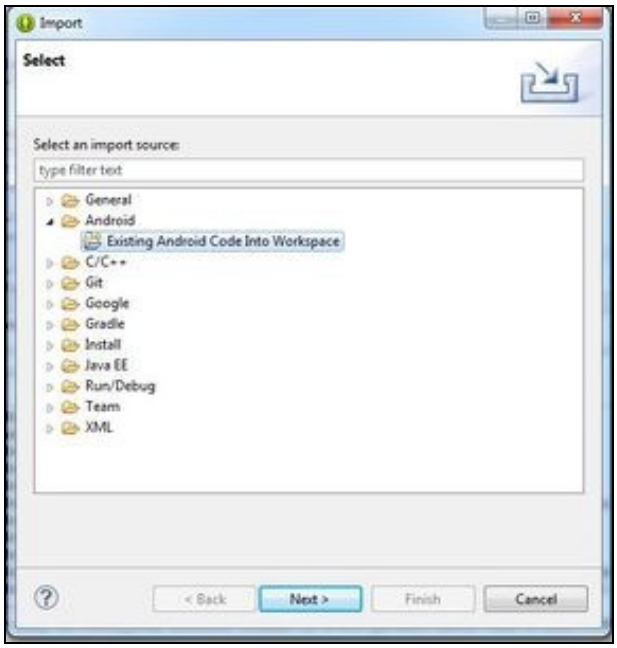
- Nota: Se usades Android Studio neste enlace indica como tedes que facer.
- Nota: Nun AVD funcionará nunha versión Android 4.2.2 ou superior cunha imaxe Google API.

Para facelo:

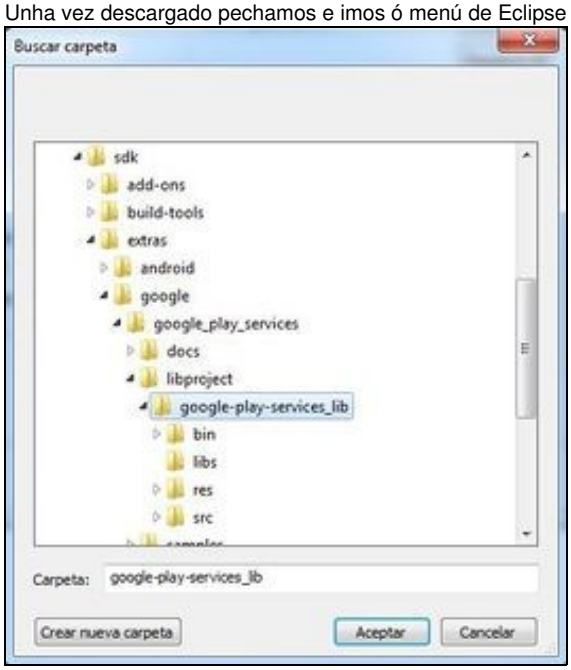
- Importación dos Google Play Services en Eclipse
- 

Primeiro abrimos o SDK Manager dende o Eclipse.
- 

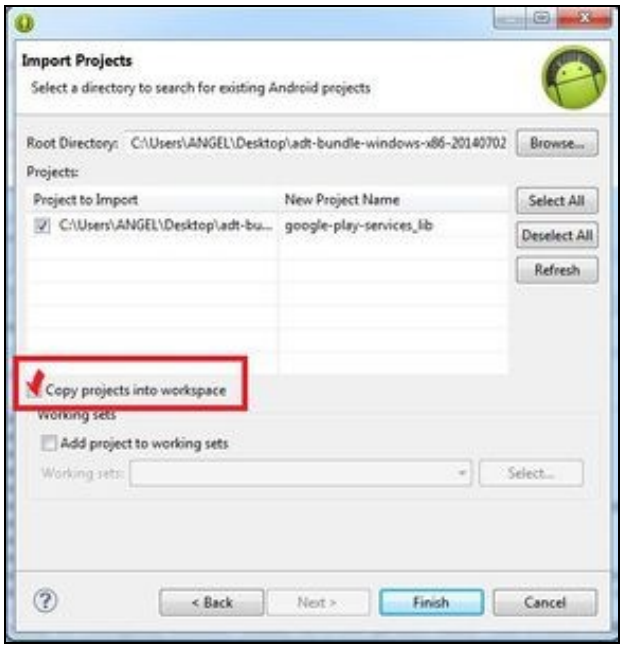
Marcamos a entrada Google Play Services (na imaxe xa está instalado).

- 

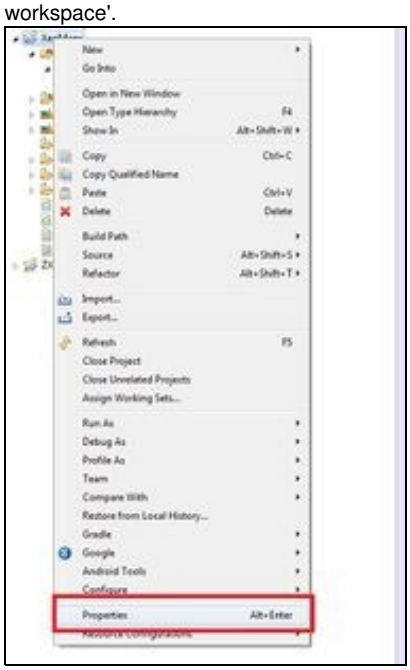
Unha vez descargado pechamos e imos ó menú de Eclipse File => Import. Escollemos a opción 'Existing Android Code into Workspace'.

- 

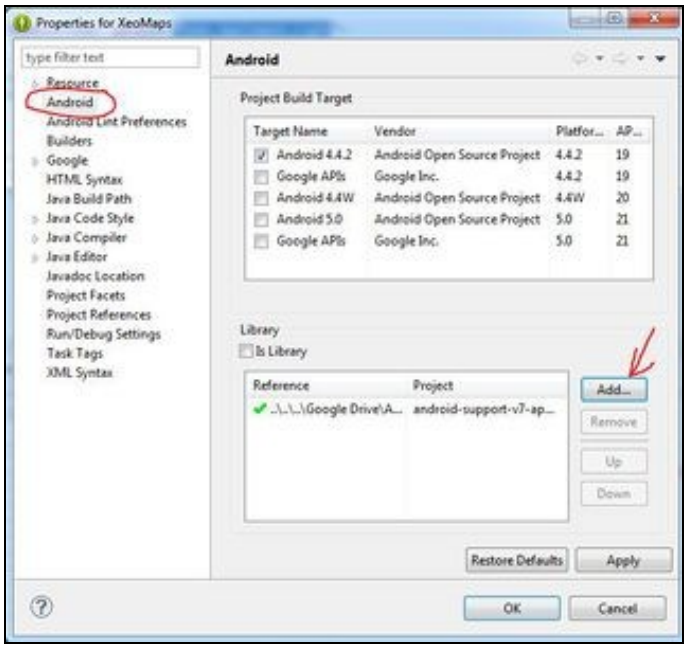
Prememos o botón Browse e imos a ruta: **<ruta-sdk>\extras\google\google_play_services\libproject\google-play-services_lib**.

- 

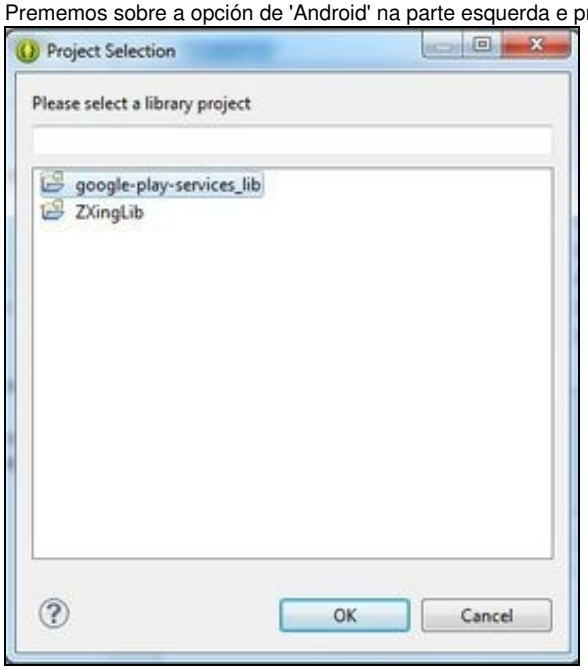
Ó seleccionar o cartafol anterior aparecerá o proxecto 'Google Play Services Lib'. Debemos seleccionar a opción de 'Copy proyect to workspace'.

- 

Agora nos situamos sobre **o noso proxecto (onde imos desenvolver a aplicación)** e prememos o botón dereito do rato escollendo a opción de **propiedades**.

- 

Prememos sobre a opción de 'Android' na parte esquerda e prememos o botón de 'Add'.

- 

Escollemos a librería Google Play Sercices.

A maiores debemos engadir o seguinte permiso no arquivo **AndroidManifiest.xml**:

- Dentro da entrada <application>:

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

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />
        ............
```

Para saber por código se un dispositivo ten o Google Play Service:

```
int resultCode =GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);

if (ConnectionResult.SUCCESS != resultCode) {

  // Neste caso ou non ten instalado o Google Play Services ou necesita actualizarse,...
}
```

En caso de erro podemos descubrir que pasa:

```
GooglePlayServicesUtil.getErrorDialog(resultCode, this, 0).show();
```

1.4 Xeración das API KEYS

Para poder utilizar un mapa de Google necesitamos xerar unha Key dende a consola de Google. Para ser máis concretos imos necesitar xerar dúas Keys, unha para facer procesos de debugger (cando estamos a desenrolar a aplicación en Eclipse) e outra xa definitiva para cando xeremos o APK que será entregado ou descargado polos usuarios da nosa aplicación.

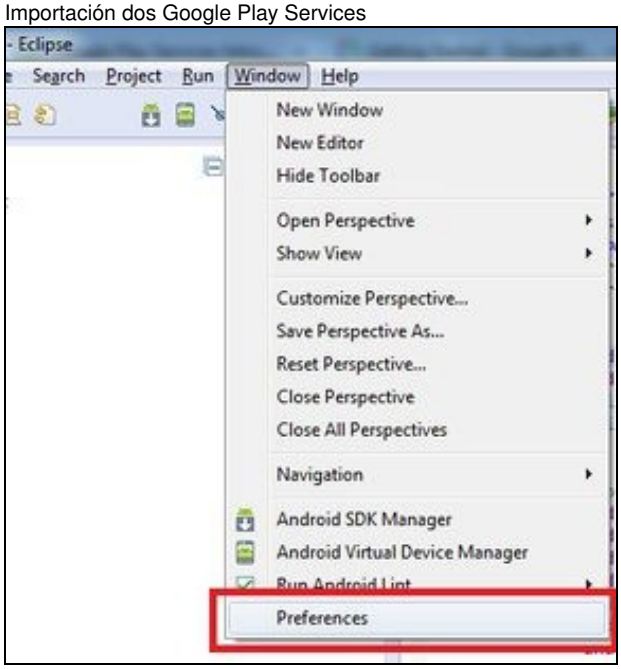
1.4.1 Obtención da pegada dixital SHA1

Un paso necesario para obter esa Key é obter a pegada dixital.

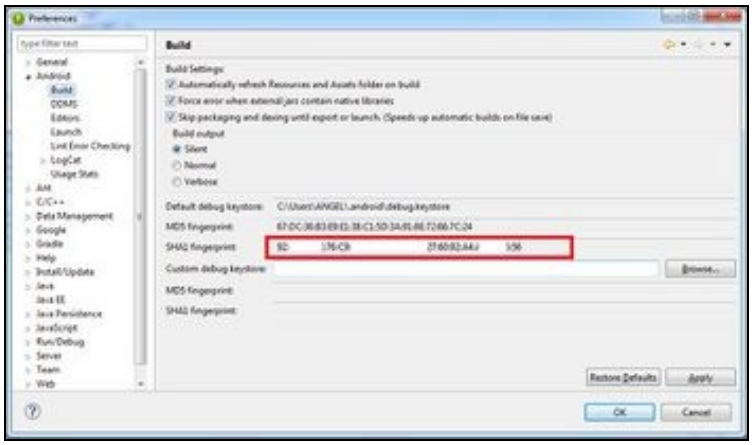
Como comentamos antes imos ter dúas, unha mentres estamos a desenrolar a aplicación e outra para cando teñamos a aplicación rematada e lista para entregar ós usuarios.

1.4.1.1 Obtención da pegada dixital SHA1 Depuración

1.4.1.1.1 Graficamente

- Importación dos Google Play Services
- 

Imos o menú Window => Preferences de Eclipse.

- 

  Copiamos a pegada SHA1.

1.4.1.1.2 Dende consola

Esta información tamén se pode obter dende unha consola ou terminal. En Windows é mellor utilizar unha consola con permisos administrativos (executar como administrador).

Debemos situarnos coa orde **cd** (se non o temos no path) no cartafol onde estea instalado o JDK e dentro deste no cartafol /bin/.

Nese cartafol se atopa o executable keytool.

- LINUX:

```
./keytool -list -v -keystore ~/.android/debug.keystore -alias androiddebugkey -storepass android -keypass android
```

- WINDOWS:

```
keytool -list -v -keystore "%USERPROFILE%\.android\debug.keystore" -alias androiddebugkey -storepass android -keypa
```

1.4.1.2 Obtención da pegada dixital SHA1 Produción

Esta a obteremos cando a aplicación estea rematada e xeremos o apk a entregar ós usuarios.

Esta parte está contemplada na Unidade de empaquetado e distribución.

1.4.2 Obtención da API KEY

Unha vez temos a pegada dixital, temos que acceder ó sitio web: https://code.google.com/apis/console/?noredirect

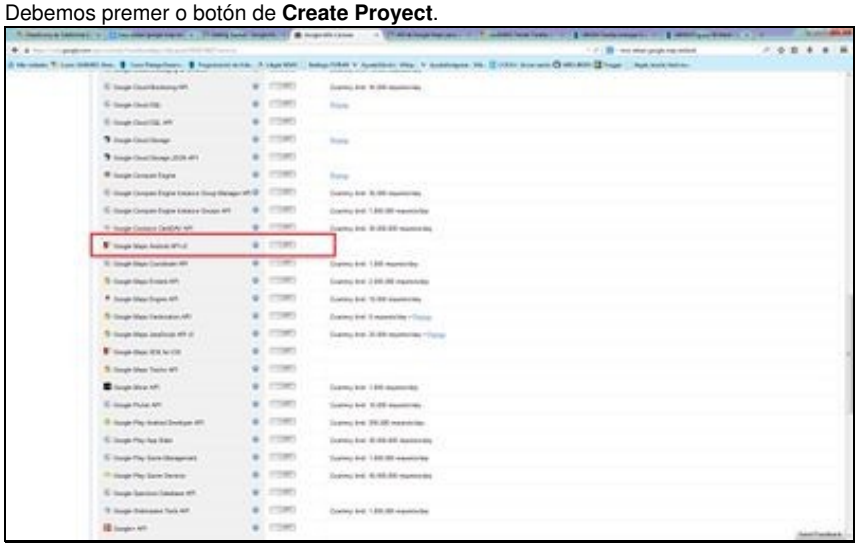
Nota: Solicitará entrar cunha conta de gmail. Se non a temos teremos que rexistrarnos e crear unha nova.

O proceso é o seguinte:

- Obtención da API KEY

- 

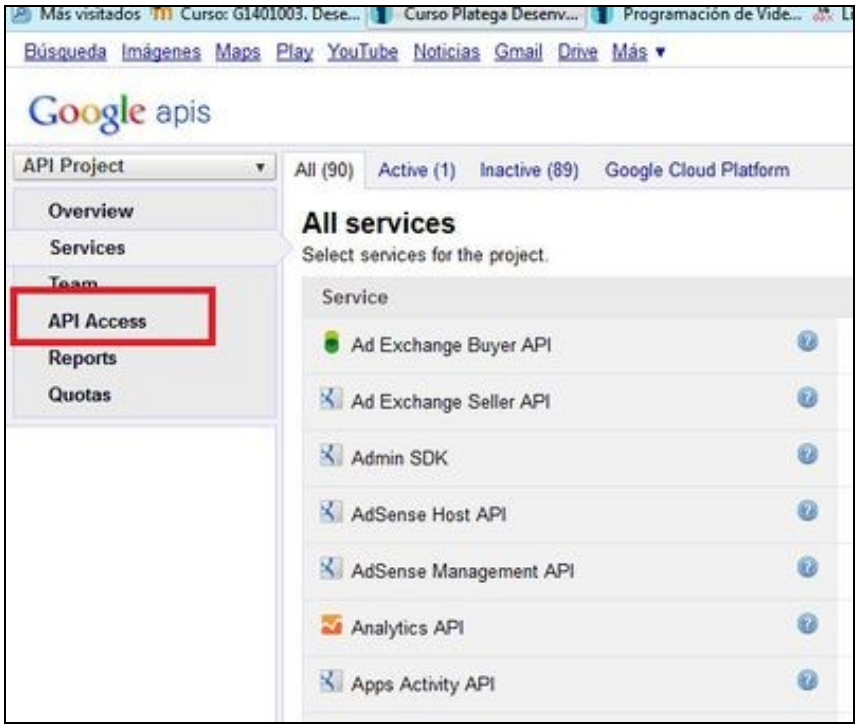
Debemos premer o botón de **Create Proyect**.

- 

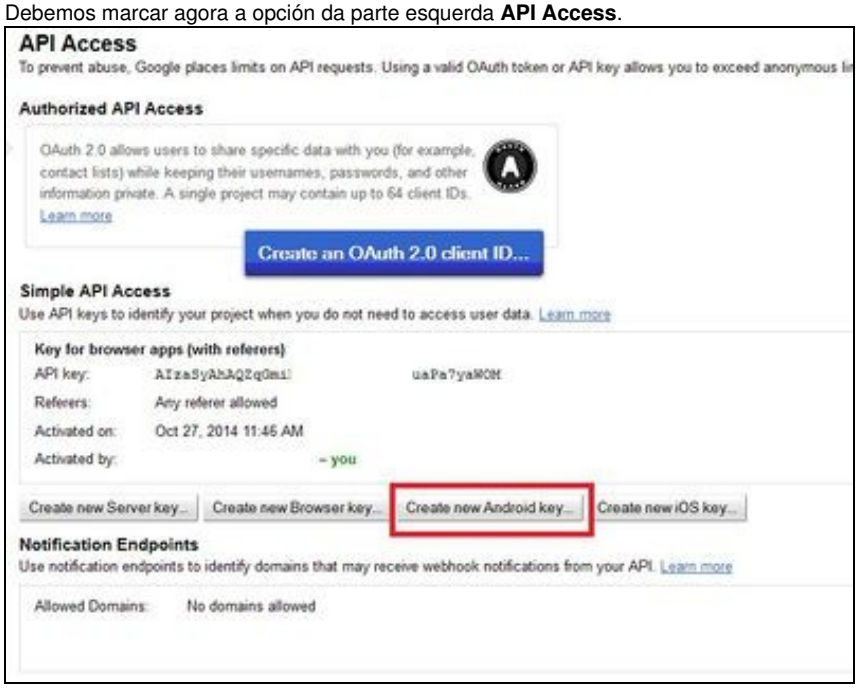
Na seguinte pantalla teremos que escoller na parte esquerda a opción de **Services** e na lista que aparece teremos que activar (premer onde pon off) o servizo **Google Maps Android API v2**.

- 

Marcamos I Agree e prememos o botón de Accept. Unha vez feito o servizo está iniciado.

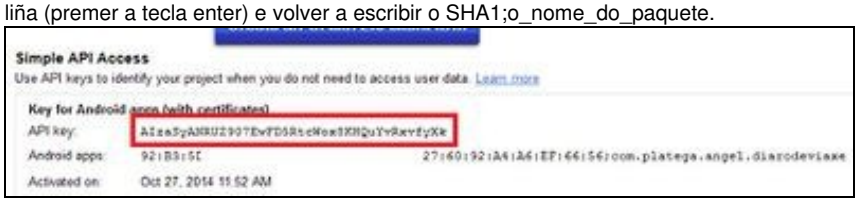
- 

Debemos marcar agora a opción da parte esquerda **API Access**.

- 

Agora debemos premer o botón **Create new Android Key**.

- 

Agora debemos copiar e pegar a pegada dixital (SHA1) obtida no punto anterior seguido por ';' e o nome do paquete do noso proxecto. Lembrar que o nome do paquete o tedes no AndroidManifiest.xml (<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="**com.platega.angel.diarodeviaxe**"). Podemos asociar varias aplicacións a mesma API KEY. Teremos que separar por unha nova liña (premer a tecla enter) e volver a escribir o SHA1;o_nome_do_paquete.
- 

Agora no listado aparece a nova key.

Nota Importante:

Lembrar que se queremos entregar a aplicación aos usuarios (xerar o APK) debemos de obter e utilizar a API KEY **a partires da SHA1 de produción**, como indicamos neste enlace.

O lóxico é que primeiro xeredes unha API KEY a partires do voso Eclipse (cando se está a facer probas). Como xa está xerada a API KEY o único que teredes que facer é editar quen ten acceso a utilizar esa api key premendo o botón 'Editar Aplicaciones Android Permitidas'. Nese intre teredes que poñer o SHA1 obtido do voso almacén de claves (obtención da SHA1).

1.5 Permisos necesarios e uso da nova API KEY

Para poder utilizar GoogleMap debemos modificar o arquivo **AndroidManifiest.xml** coas seguintes entradas:

- **Xusto antes da etiqueta </application>:**

```
...........
    <meta-data
       android:name="com.google.android.maps.v2.API_KEY"
       android:value="API_KEY"/>
</application>
...........
```

Tendo que substituír en value API_KEY pola API KEY xerada no paso anterior.

- Debemos engadir os seguintes permisos:

```
<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" />
```

- Os seguintes permisos non son obrigatorios pero se recomendan utilizalos:

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

O primeiro serve para permitir que se utilice o WIFI ou antenas móbiles para determinar a localización.
O segundo serve para permitir que se utilice o GPS para determinar a localización.

- É necesario que a tarxeta gráfica soporte OPEN GL 2.0 para a renderización do mapa.

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

1.6 Renderizado do MAPA

Debemos crear un layout para a activity no que se visualizará o mapa.

Aquí temos varias opcións:

- Podemos facer que o mapa ocupe todo o layout:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.MapFragment"/>
```

- Podemos facer que o mapa forme parte dun layout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
>

<fragment
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.MapFragment"/>

</RelativeLayout>
```

- Unha vez definido o layout creamos a activity que o carga:

```
import android.app.Activity;
import android.os.Bundle;

public class UD6_02_Mapa extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.layout_anteriormente_definido);
}
}
```

Se facemos todos os pasos indicados teremos como resultado algo parecido a isto:

1.6.1 Versións da API

Como podemos comprobar o mapa utiliza unha clase MapFragment para visualizarse.

Esta clase foi introducida a partires da API 12 (Android 3.1). Se necesitamos desenrolar unha aplicación para unha versión anterior teremos que modificar o layout e poñer o seguinte:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.SupportMapFragment"/>
/>
```

- Neste caso a activity debe ser unha subclase da clase FragmentActivity que se atopa na librería de compatibilidade v4:

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class UD6_02_Mapa  extends FragmentActivity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_ud6_02_mapa);
}
}
```

1.7 Manexo do Mapa

Unha vez temos o mapa no layout podemos facer referencia a el dende a activity e modificar propiedades, facer zoom, mover a cámara...

Faremos uso da clase GoogleMap:

- Para referenciar o mapa:

```
import com.google.android.gms.maps.GoogleMap;

private GoogleMap googleMap;

googleMap = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();
```

- Se necesitamos que o GoogleMap funcione en versións anteriores á API 12 (Android 3.1) teremos que refencialo desta forma, facendo uso da librería de compatibilidade v4:

```
import com.google.android.gms.maps.GoogleMap;

private GoogleMap googleMap;

googleMap = ((SupportMapFragment) getSupportFragmentManager().findFragmentById(R.id.map)).getMap();
```

Entre os métodos que podemos empregar:

- setMapType(TIPO_MAPA): Cambio o tipo de mapa, sendo TIPO_MAPA:
  - ◊ GoogleMap.MAP_TYPE_TERRAIN
  - ◊ GoogleMap.MAP_TYPE_NORMAL
  - ◊ GoogleMap.MAP_TYPE_SATELLITE
  - ◊ GoogleMap.MAP_TYPE_HYBRID

- moveCamera(TIPO_MOVEMENTO): Move a cámara directamente.
- animateCamera(TIPO_MOVEMENTO): Move a cámara cunha animación.

  sendo TIPO_MOVEMENTO:

  - CameraUpdateFactory.zoomIn(): Aumenta en 1 o zoom.
  - CameraUpdateFactory.zoomOut(): Diminúe en 1 o zoom.
  - CameraUpdateFactory.zoomTo(nivel_de_zoom): Nivel de zoom entre 2 e 21.

  - CameraUpdateFactory.newLatLng(lat, long): Nova lonxitude e latitude.
  - CameraUpdateFactory.newLatLngZoom(lat, long, zoom): Nova lonxitude e latitude cun zoom determinado.

  - CameraUpdateFactory.scrollBy(scrollHorizontal, scrollVertical): Fai un movemento en scroll desprezando o mapa o número de pixeles indicados.

  Nota: Para mover a cámara ou cambiar o zoom faremos uso da clase CameraUpdateFactory e os seus métodos de clase.

  Exemplo de código:

```
LatLng pos = new LatLng(latitude, lonxitude);
googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(pos,15));
```

Importante: Fixarse como para manexar lonxitudes-latitudes temos que facer uso da clase LatLng.

- getCameraPosition(): Devolve un obxecto da clase CameraPosition a posición da cámara. Dentro desta clase podemos chamar á propiedade target que nos devolve un obxecto da clase LatLng que representa a lonxitude e latitude da posición da cámara.

  Exemplo de código:

```
CameraPosition posCam = googleMap.getCameraPosition();
LatLng posicion = posCam.target;
```

Tamén podemos obter o zoom, orientación e ángulo accedendo ás propiedades zoom, bearing e tilt.

1.8 Marcas no Mapa

Outra das aplicacións que pode ter o uso de GoogleMap é indicar mediante iconas, posicións concretas do mapa no que alberguemos certa información.

Isto o conseguimos coas clases Marker e MarkerOptions.

- MarkerOptions: Crea unha marca nova no mapa.

A forma de agregala sería:

```
googleMap.addMarker(new MarkerOptions()
.position(googleMap.getCameraPosition().target)
.title("TI")
.snippet("Ti nesta posición:"+ googleMap.getCameraPosition().target.toString())
.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_launcher)));
```

Onde:

◊ position: Posición da marca.
◊ title: Texto que aparece na marca.
◊ icon: Icona asociada á marca.
◊ sniper: Texto que aparece cando pulsamos sobre a marca no mapa.

A chamada ó método addMarker nos devolve o obxecto da clase Marker:

- Marker: Representa a marca no mapa.

```
Marker marca;
marca = googleMap.addMarker(new MarkerOptions()
.position(googleMap.getCameraPosition().target)
.title("TI")
.snippet("Ti nesta posición:"+ googleMap.getCameraPosition().target.toString())
.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_launcher)));
```

A través deste obxecto podemos obter:

◊ getPosition(): A posición (obxecto da clase LatLng).
◊ getRotation(): A súa rotación.
◊ getTitle(): O seu título.

.....

◊ remove(): eliminamos a marca do mapa.
◊ setVisible(boolean): Indicamos se queremos que sexa visible a marca no mapa.

1.9 Liñas no Mapa

Algunhas veces pode sernos de utilizade 'unir' diferentes marcas no mapa (para indicar unha ruta por exemplo).

Para facelo debemos utilizar a clase PolygonOptions.

Esta clase dispón do método add no que se lle pasa a posición a engadir (en forma de obxecto da clase LatLng).

Cando se engade un obxecto desta clase ó mapa, este une os puntos.

Por exemplo:

```
PolygonOptions polOpt = new PolygonOptions();
polOpt.add(new LatLng(10,10));
polOpt.add(new LatLng(11,11));
polOpt.add(new LatLng(12,12));
polOpt.strokeColor(Color.BLUE);

googleMap.clear();
googleMap.addPolygon(polOpt);
```

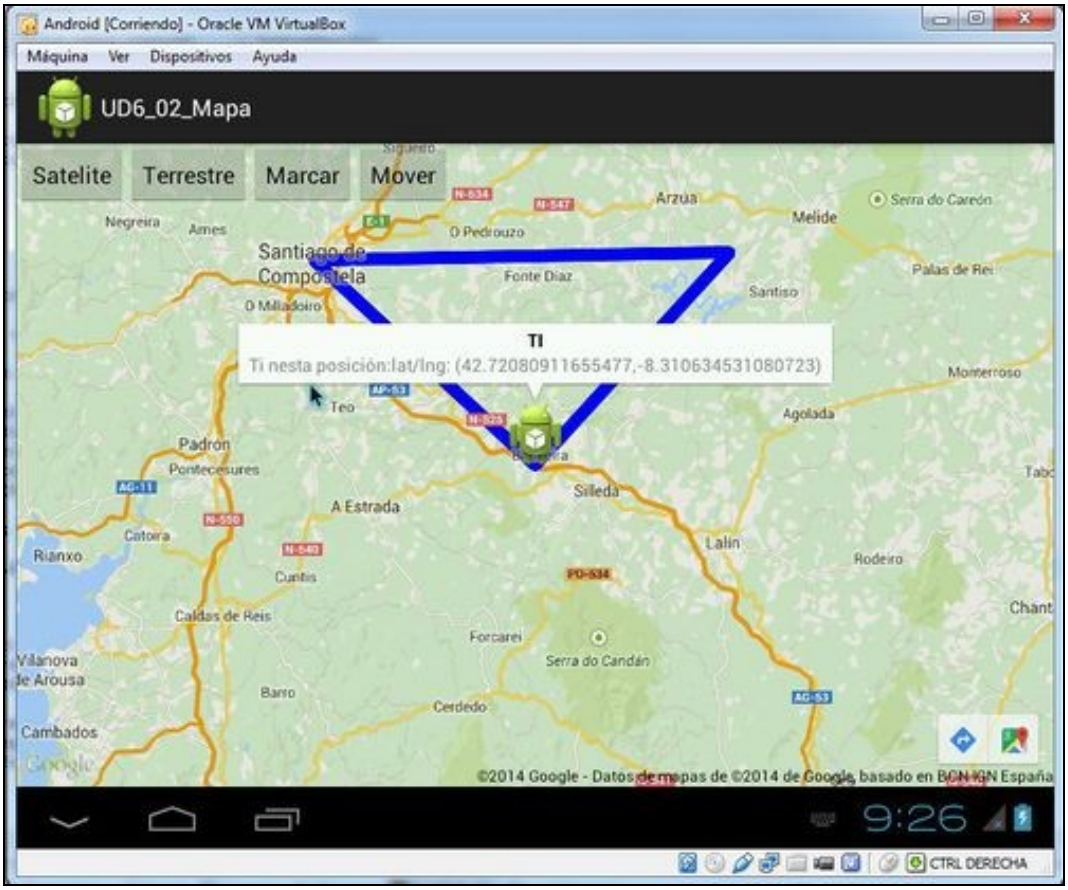
- Liña 5: Establece unha cor para a liña que vai unir os diferentes puntos.
- Liña 7: Limpa o mapa de marcas e liñas previamente debuxadas.
- Liña 8: Engade ó mapa o conxunto de liñas.

1.10 Caso Práctico

O obxectivo desta práctica é crear unha pantalla cun mapa de google.

Poderemos cambiar de tipo de mapa (Satélite / Terrestre), movernos ó centro do mapa e gardar a posición e ir uníndoas cun polígono.

O aspecto desta práctica é o seguinte:



- Como funciona a práctica:

  A nosa icona sempre se visualizará no centro do mapa. Esta será a posición que poderemos gardar como 'puntos de paso' e que despois se unirán formando unha ruta.
  Na parte superior temos:

  Botón Satélite: Cambia o aspecto a modo satélite.
  Botón Terrestre: Cambia o aspecto a modo terrestre.
  Botón Marcar: Garda nun array a posición da icona. Unha vez temos gardados máis de dous, crea un polígono unindo os puntos.
  Botón Mover: Move a icona ó centro do mapa. Para mover o mapa temos que premer sobre o mapa e sen soltar, arrastrar.

  Ó premer sobre a icona informa da posición e amosa un texto.

1.10.1 IMPORTANTE: Aspectos a ter en conta para que funcione este práctica

- Se estamos a utilizar un AVD temos que utilizar un que sexa 'Google API 21' e cambiar as propiedades do proxecto tal como indicamos neste punto: Aclaración sobre o dispositivo virtual android (AVD)

- Necesitaremos xerar unha API KEY para o voso eclipse e proxecto como vimos neste punto: Obtención da API KEY.

- Unha vez obtida debemos de escribila no sitio correspondente do arquivo AndroidManifiest.xml

1.10.2 Creamos a activity

- Nome do proxecto: **UD6_02_Mapa**
- Nome da activity: **UD6_02_Mapa.java**

Código do layout xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"

>

<fragment
          android:id="@+id/map"
          android:layout_width="match_parent"
          android:layout_height="match_parent"
          android:name="com.google.android.gms.maps.MapFragment"/>

<Button
    android:id="@+id/UD6_02_btnSatelite"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:text="Satelite" />

<Button
    android:id="@+id/UD6_02_btnTerrestre"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_toRightOf="@+id/UD6_02_btnSatelite"
    android:text="Terrestre" />

<Button
    android:id="@+id/UD6_02_btnMarcar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_toRightOf="@+id/UD6_02_btnTerrestre"
    android:text="Marcar" />

<Button
    android:id="@+id/UD6_02_btnMover"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentTop="true"
    android:layout_toRightOf="@+id/UD6_02_btnMarcar"
    android:text="Mover" />

</RelativeLayout>
```

Código da clase UD6_02_Mapa

Obxectivo: Traballar cun mapa de Google Map utilizando marcas.

```
import java.util.ArrayList;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
```

```
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolygonOptions;

public class UD6_02_Mapa extends Activity {

private GoogleMap googleMap;
private Marker marcaActual;
private ArrayList<LatLng>marcas;

private void xestionarEventos(){

Button btnSatelite = (Button)findViewById(R.id.UD6_02_btnSatelite);
btnSatelite.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub

googleMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);

}
});
Button btnTerrestre = (Button)findViewById(R.id.UD6_02_btnTerrestre);
btnTerrestre.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub

googleMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);

}
});

Button btnMover = (Button) findViewById(R.id.UD6_02_btnMover);
btnMover.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub

if (marcaActual != null)
marcaActual.remove();

marcaActual = googleMap.addMarker(new MarkerOptions()
.position(googleMap.getCameraPosition().target)
.title("TI")
.snippet(
"Ti nesta posición:"
+ googleMap.getCameraPosition().target
.toString())
.icon(BitmapDescriptorFactory
.fromResource(R.drawable.ic_launcher)));

}
});

Button btnMarcar = (Button)findViewById(R.id.UD6_02_btnMarcar);
btnMarcar.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub

marcas.add(googleMap.getCameraPosition().target);

PolygonOptions polOpt = new PolygonOptions();
for (LatLng lugar : marcas){
polOpt.add(lugar);
}
polOpt.strokeColor(Color.BLUE);
```

```
googleMap.clear();
googleMap.addPolygon(polOpt);

marcaActual = googleMap.addMarker(new MarkerOptions()
.position(googleMap.getCameraPosition().target)
.title("TI")
.snippet(
"Ti nesta posición:"
+ googleMap.getCameraPosition().target
.toString())
.icon(BitmapDescriptorFactory
.fromResource(R.drawable.ic_launcher)));

}
});

}

private void prepararMapa(){

googleMap = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();

if (googleMap==null){
Toast.makeText(this, "ERRO O OBTER O MAPA", Toast.LENGTH_LONG).show();
finish();
}

LatLng pos = new LatLng(42.879985, -8.544855);// Posición de Santiago de Compostela

marcaActual = googleMap.addMarker(new MarkerOptions()
.position(pos)
.title("TI")
.snippet("Ti nesta posición:" + pos.toString())
.icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_launcher))
);

googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(marcaActual.getPosition(),16));
}

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_ud6_02__mapa);

marcas = new ArrayList<LatLng>();
prepararMapa();
xestionarEventos();
}
}
```

- Liña 22: Definimos o mapa de google.
- Liña 23: A marca que representa a nosa posición. É necesaria xa que se queremos movela temos que primeiro eliminala e despois volvela a crear.
- Liña 24: Definimos o array que vai representar o conxunta de puntos que temos gardados.

- Liña 35: Cambiamos o modo a Satélite ó premer o botón.
- Liña 46: Cambiamos o modo a Terrestre ó premer o botón.
- Liñas 58-69: Xestionamos o evento Click sobre o botón mover.

    ◊ Liñas 58-59: Eliminamos a icona do mapa.
    ◊ Liñas 61-69: Agregamos unha nova icona (marca) na posición actual da cámara.

- Liñas 81-100: Xestionamos o evento Click sobre o botón marcar.

    ◊ Liña 81: Agregamos a posición actual da cámara ó array de marcas.
    ◊ Liñas 83-86: Creamos un PolygonOptions e agregamos todas as marcas gardadas no array.
    ◊ Liña 87: Establecemos de cor azul a liña que une os puntos.
    ◊ Liña 89: Limpa o mapa.

◊ Liña 90: Agrega o PolygonOptions para debuxar as liñas entre os puntos.
◊ Liñas 92-100: Agregamos unha nova icona (marca) na posición actual da cámara.

- Liñas 107-126: Obtén unha referencia ó mapa e crea unha marca (icona) en Santiago de Compostela.

◊ Liña 109: Obtén unha referencia ó mapa.
◊ Liñas 116-123: Crea unha marca (icona) en Santiago de Compostela.
◊ Liña 125: Move a cámara do mapa á posición da marca cun zoom de 15.

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2014).