La Clase `MapView`.

CONCEPTOS.

La API de Google Location Services, parte de servicios de Google Play, ofrece un marco más potente, de alto nivel que automatiza tareas como la elección del proveedor y la administración de energía. Los servicios de Localización también proporcionan nuevas características, como la detección de la actividad que no está disponible en la API. Los desarrolladores que utilizan la API, así como los que apenas están agregando la ubicación a las aplicaciones, deben considerar seriamente el uso de la API de servicios de localización.

La localización y las aplicaciones basadas en mapas ofrecen una experiencia atractiva en los dispositivos móviles. Se pueden construir estas capacidades en la aplicación utilizando las clases del paquete `android.location` y la API de Google `Maps` para Android.

Servicios de localización

Android le da a las aplicaciones el acceso a los servicios de localización, compatible con el dispositivo, a través de las clases del paquete `android.location`. El componente central de la ubicación es el servicio del sistema `LocationManager`, que proporciona APIs para determinar la ubicación y el manejo del dispositivo subyacente (si está disponible).

Al igual que con otros servicios del sistema, no se crean instancias de un `LocationManager` directamente. Más bien, se solicita una instancia del sistema llamando a `getSystemService(Context.LOCATION_SERVICE)`. El método devuelve un identificador a una nueva instancia de `LocationManager`.

Una vez que la aplicación tiene un `LocationManager`, su aplicación es capaz de hacer tres cosas:

- Consultar la lista de todos los `LocationProviders` de la última ubicación conocida del usuario.
- Registrar/eliminar el registro de actualizaciones periódicas de la ubicación actual del usuario desde un proveedor de ubicación (especificada ya sea por criterios o nombre).
- Registrar/eliminar el registro de un intento determinado por dispararse si el dispositivo está dentro de una proximidad dada (especificada por el radio, en metros) de una determinada latitud/longitud.

API de Google Maps para Android

Con la API de Google Maps para Android, se pueden añadir mapas a la aplicación, que se basan en datos de mapas de Google. La API se encarga de automatizar el acceso a los servidores de Google Maps, la descarga de datos, visualización del mapa, y digitado de los gestos en el mapa. También se pueden utilizar llamadas a la API para agregar marcadores, polígonos y superposiciones, y para cambiar la vista del usuario de un mapa de una zona en particular.

La clase de la clave de la API de Google Maps para Android es `MapView`. Un `MapView` muestra un mapa con los datos obtenidos del servicio de Google Maps. Cuando el `MapView` tiene el foco, se captarán las digitaciones y gestos táctiles para alejar o acercar el mapa de forma automática, incluyendo el manejo de las solicitudes de red para secciones de mapas adicionales. También proporciona todos los elementos necesarios para la interfaz de usuario para controlar el mapa. La aplicación también puede utilizar los métodos de clase `MapView` para controlar el mapa mediante programación y extraer una serie de superposiciones en la parte superior del mapa.

Las API de Google Maps para Android no están incluidos en la plataforma Android, pero están disponibles en cualquier dispositivo en la tienda Play Google con Android 2.2, o superior, a través de servicios de Google Play.

Para integrar Google Maps en la aplicación, es necesario instalar las bibliotecas de los servicios de Google Play para el SDK de Android.

La clase pública **`MapView`** hereda de **`FrameLayout`**

Es un View que muestra un mapa (con datos obtenidos del servicio de Google Maps). Cuando se conecta, se captan las digitaciones y gestos táctiles para mover el mapa.

Los usuarios de esta clase deben remitir todos los métodos de ciclo de vida de la `Activity` o `Fragment` que contiene este `View` a los correspondientes en esta clase:

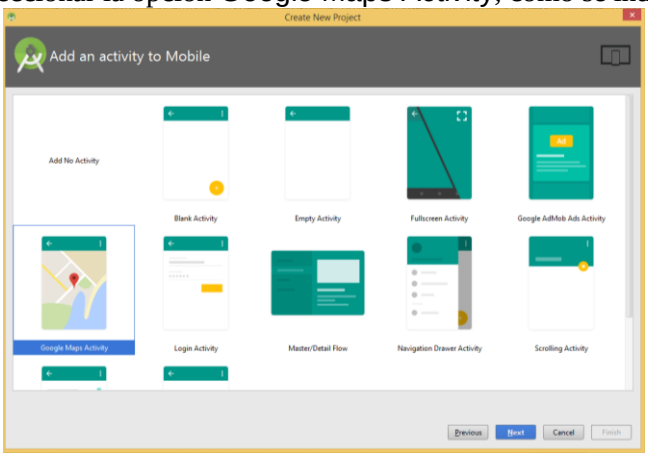
- `onCreate(Bundle)`
- `onResume()`
- `onPause()`
- `onDestroy()`
- `onSaveInstanceState()`
- `onLowMemory()`

El GoogleMap debe ser adquirido mediante `getMapAsync(OnMapReadyCallback)`. El `MapView` inicializa automáticamente el sistema de mapas y la vista.
Nota: Se recomienda no agregar hijos a este View.

DESARROLLO.

EJEMPLO 1.

Paso 1. Al abrir el IDE de Android Studio, crear un nuevo proyecto en la ventana **Configure your new project** y en el campo Application name: ingresar el nombre `Mapas`, enseguida, clic en Next. En la siguiente ventana **Select the form factors your app will run**, verificar la selección de Phone and Tablet, enseguida clic en Next. En la siguiente ventana **Add an activity to Mobile**, seleccionar la opción Google Maps Activity, como se indica en la imagen siguiente:



Enseguida, clic en Next. Por último, clic en Finish.

Paso 2. En la carpeta java/com.example.mipaquete, abrir y verificar el código del archivo predeterminado MapsActivity.java, según el código siguiente:

```
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import com.google.android.gms.maps.*;
import com.google.android.gms.maps.model.*;
/*import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
```

```
import com.google.android.gms.maps.model.MarkerOptions;*/
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {
    private GoogleMap mMap;
    @Override
    protected void onCreate(Bundle b) {
        super.onCreate(b);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be
used.
        SupportMapFragment mapFragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }
    /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In
this case,
     * we just add a marker near Sydney, Australia.
     * If Google Play services is not installed on the device, the user will be prompted
to install
     * it inside the SupportMapFragment. This method will only be triggered once the user
has
     * installed Google Play services and returned to the app.
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;
        // Add a marker in Sydney and move the camera
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```

Paso 3. En la carpeta res/layout, abrir y verificar el código del archivo predeterminado activity_maps.xml, según el código siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"

    android:id="@+id/map"
    android:name="com.google.android.gms.maps.SupportMapFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
     />
```

Paso 4. En la carpeta res/values, abrir y verificar el código del archivo predeterminado google_maps_api.xml, según el código siguiente. Observar las líneas marcadas con letras negritas, para descargar una clave válida, el sitio web de obtención de la clave y el nombre de la clave, respectivamente. Esto anterior, se indicará con detalle en las secciones de ejercicios:

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.
```

```
    To get one, follow this link, follow the directions and press "Create" at the end:


https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=04:86:1F:AA:54:9C:E7:BD:80:56:1C:D3:41:6C:A1:C5:3F:F6:CB:E2%3Bcom.example.escom.mapas

    You can also add your credentials to an existing key, using this line:
    04:86:1F:AA:54:9C:E7:BD:80:56:1C:D3:41:6C:A1:C5:3F:F6:CB:E2;com.example.escom.mapas

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
        AIzaSyCE7m4NdIbhVg_bl3WQLUlswqJd3LUIV9w
    </string>
</resources>
```

Paso 5. En la carpeta app/manifests, abrir y verificar el código del archivo predeterminado AndroidManifest.xml, según el código siguiente:

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

    <!--
         The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but you must specify either coarse or fine
         location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

    <application
        android:fullBackupContent="false"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        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"
```

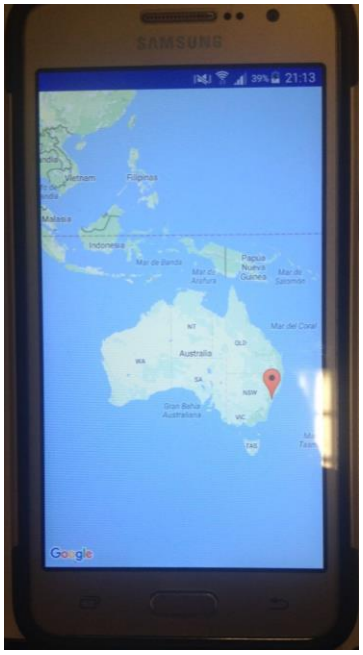
```
            android:value="@string/google_maps_key" />

        <activity
            android:name="com.example.escom.mapas.MapsActivity"
            android:label="@string/title_activity_maps">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Paso 6. Por último, ejecutar la aplicación en un dispositivo real. Si el dispositivo posee una conexión a Internet, la aplicación mostrará un mapa con la ubicación inicial en Sydney, Australia; enseguida se puede navegar en el mapa y aplicar varios gestos digitales para cambiar la ubicación predeterminada.



EJERCICIOS.

SECCIÓN I.

Parte I.

Verificar la instalación del SDK de Google Play Services; en caso contrario, agregar el paquete de Google Play Services a Android Studio.

Parte 2:

- Crear un proyecto de Google Maps
- Seleccionar Start a new Android Studio project.
- Ingresar el nombre de la aplicación, el dominio y la ubicación del proyecto, si es necesario. Clic en Next.
- Seleccionar Phone and Tablet. Clic en Next.
- En el cuadro de diálogo Add an activity to Mobile, seleccionar Google Maps Activity. Clic en Next.
- Ingresar el nombre de la actividad, diseño y título. Clic en Finish.

En el IDE de Android Studio abrir los archivos google_maps_api.xml y MapsActivity.java en el editor. El archivo google_maps_api.xml muestra las instrucciones para descargar una clave desde el sitio Google Maps API.

Parte 3.

- Esta aplicación requiere una clave de API para utilizar los servicios de Google Maps. La clave es gratuita y se puede utilizar con otras aplicaciones que utilicen la API de Google Maps Android.
- Localizar y abrir el archivo google_maps_api.xml en el navegador.

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.
    To get one, follow this link, follow the directions and press "Create" at the end:
https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=
CLIENT_SIDE_ANDROID&r=04:86:1F:AA:54:9C:E7:BD:80:56:1C:D3:41:6C:A1:C5:3F:F6:CB:E2%3Bcom.e
xample.escom.mapas
    You can also add your credentials to an existing key, using this line:
    04:86:1F:AA:54:9C:E7:BD:80:56:1C:D3:41:6C:A1:C5:3F:F6:CB:E2;com.example.escom.mapas
    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key
    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
        AIzaSyCE7m4NdIbhVg_bl3WQLUlswqJd3LUIV9w
    </string>
</resources>
```

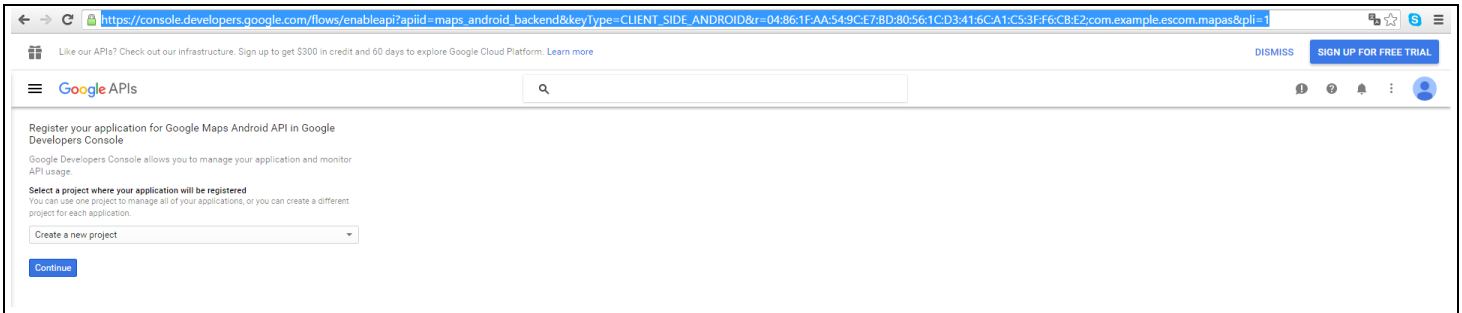
- En el contenido del archivo localizar la siguiente línea comentada:

```
https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SID
E_ANDROID&r=04:86:1F:AA:54:9C:E7:BD:80:56:1C:D3:41:6C:A1:C5:3F:F6:CB:E2%3Bcom.example.escom.mapas
```

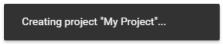
- Abrir el navegador, copiar el vínculo anterior y pegarlo en el campo de la URL. El vínculo permite el acceso a Google Developers Console.
- Seguir las instrucciones de la Parte 4.

Parte 4.

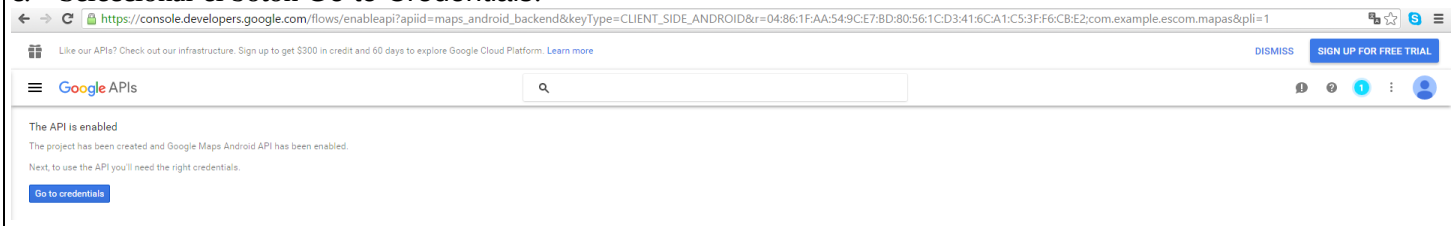
a. Abrir el navegador y pegar la URL en el campo de texto:



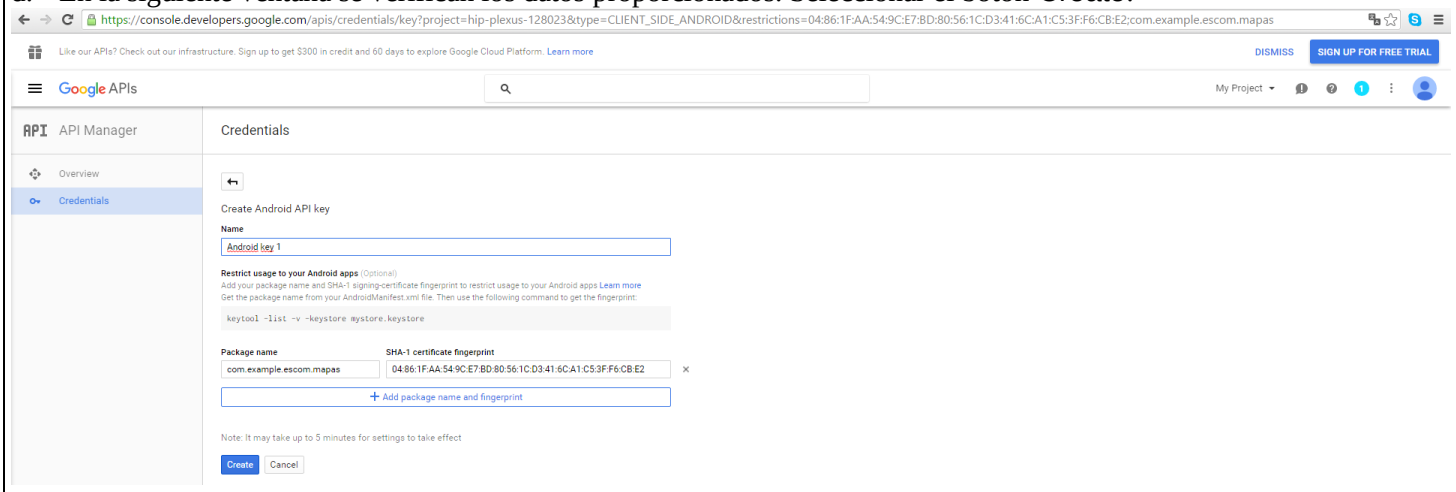
b. Seleccionar el botón Continue. En la sección inferior se muestra un mensaje que indica la creación del nuevo proyecto:



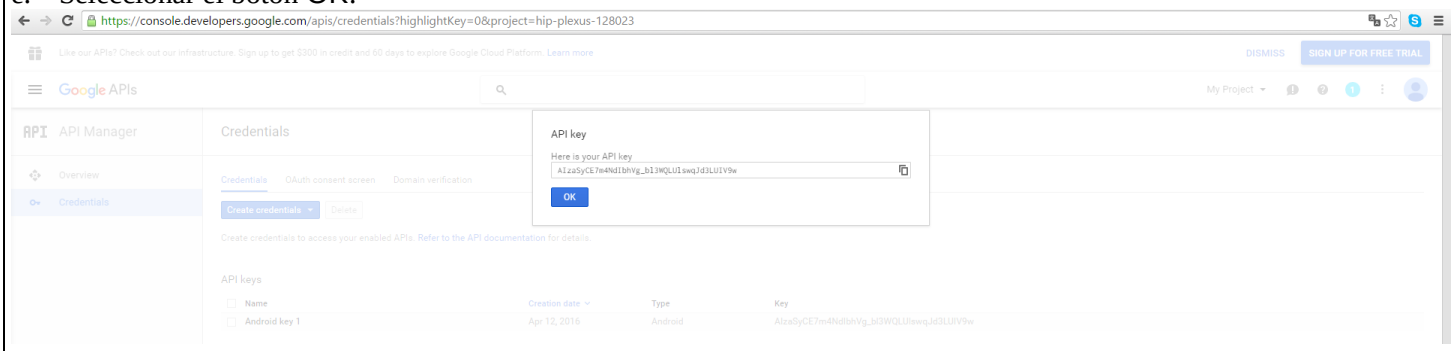
c. Seleccionar el botón Go to Credentials:



d. En la siguiente ventana se verifican los datos proporcionados. Seleccionar el botón Create:

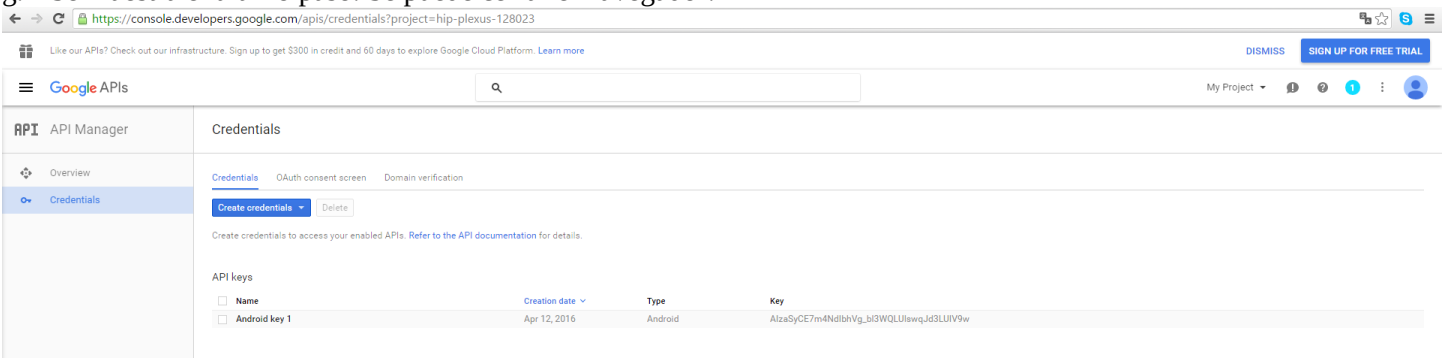


e. Seleccionar el botón OK:



f. Se muestra la API Key. Seleccionar el botón OK:

g. Se muestra el último paso. Se puede cerrar el navegador:



h. La API Key **es: `AIzaSyCE7m4NdIbhVg_bl3WQLUlswqJd3LUIV9w`**

La clave API se debe pegar entre los elementos `<string> </string>` del archivo google_maps_api.xml:

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

Debiendo quedar de la siguiente forma:

```
<string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
AIzaSyCE7m4NdIbhVg_bl3WQLUlswqJd3LUIV9w
</string>
```

Si surgiera alguna duda para la obtención de la clave API, se puede consultar el sitio siguiente:
https://developers.google.com/maps/documentation/android-api/signup
https://developers.google.com/maps/documentation/android-api/signup#obtener_una_clave_de_la_android_api

Parte 5.

- Conectar un dispositivo Android a la computadora.
- Digitar clic en Run para ejecutar la aplicación.
- Seleccionar el dispositivo Android conectado a la computadora.
- Se muestra un mapa con un marcador sobre Sídney, Australia, debido a lo siguiente:

```
LatLng sydney = new LatLng(-34, 151);
mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
```

Nota:

- Una forma alternativa es utilizar las credenciales incluidas en el archivo google_maps_api.xml en el IDE de Android Studio.
- Copiar las credenciales desde el archivo google_maps_api.xml:

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.
    To get one, follow this link, follow the directions and press "Create" at the end:
```

```
https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=
CLIENT_SIDE_ANDROID&r=04:86:1F:AA:54:9C:E7:BD:80:56:1C:D3:41:6C:A1:C5:3F:F6:CB:E2%3Bcom.e
xample.escom.mapas
    You can also add your credentials to an existing key, using this line:
    04:86:1F:AA:54:9C:E7:BD:80:56:1C:D3:41:6C:A1:C5:3F:F6:CB:E2;com.example.escom.mapas
    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key
    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">
        AIzaSyCE7m4NdIbhVg_bl3WQLUlswqJd3LUIV9w
    </string>
</resources>
```

- Ingresar a Google Developers Console en el navegador.
- Utilizar las credenciales anteriores para agregar la aplicación a una clave API existente o crear una nueva.

SECCIÓN II.

Uso de `MapView`.

Cuando se agrega un `MapView` la aplicación se muestra en el modo de mapa normal pero también se puede cambiar a vista de satélite, marcar las zonas con `StreetView` y la información del tráfico con los métodos siguientes:

```
setSatellite(true)
setStreetView(true)
setTraffic(true)
isSatellite()
isStreetView()
isTraffic()
```

a. El uso de la aplicación se enriquece con la inclusión de un botón para vista normal y vista de satélite:

```
private Button btnSatelite = null;
      :
```

```
btnSatelite = (Button)findViewById(R.id.BtnSatelite);
      :
btnSatelite.setOnClickListener(new OnClickListener() {
      public void onClick(View arg0) {
      if(mapa.isSatellite())
      mapa.setSatellite(false);
      else
      mapa.setSatellite(true);
      }
});
```

a. Probando el botón se muestra la vista de satélite:

Para el zoom, se invoca al método `setBuiltInZoomControls()` que permite mostrar sus controles, así:

```
mapa.setBuiltInZoomControls(true);
```

Para conocer las coordenadas geográficas actuales en el mapa se invocan los métodos `getMapCenter()` y `getZoomLevel()`:

```
GeoPoint loc = mapa.getMapCenter();
int lat = loc.getLatitudeE6(); // latitud y longitud en microgrados (grados * 1E6)
int lon = loc.getLongitudeE6();
int zoom = mapa.getZoomLevel(); // el zoom tiene un valor entre 1 y 21
```

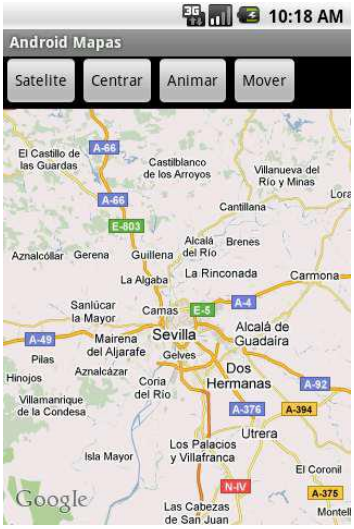
Con el método `getController()` se accede al controlador del mapa que regresa un objeto `MapController` para modificar los datos y con sus métodos `setCenter()` y `setZoom()` se indican las coordenadas centrales del mapa y el zoom.

c. Incluir un nuevo botón para centrar el mapa sobre un punto determinado, por ejemplo Sevilla, y se aplica un nivel de zoom de 10:

```
private Button btnCentrar = null;
private MapController controlMapa = null;
      :
btnCentrar = (Button)findViewById(R.id.BtnCentrar);
      :
controlMapa = mapa.getController();
      :
btnCentrar.setOnClickListener(new OnClickListener() {
```

```
      public void onClick(View arg0) {
      Double latitud = 37.40*1E6;
      Double longitud = -5.99*1E6;
      GeoPoint loc = new GeoPoint(latitud.intValue(), longitud.intValue());
      controlMapa.setCenter(loc);
      controlMapa.setZoom(10);
      }
});
```

d. Se ejecuta de nuevo la aplicación para probar los nuevos cambios:



Para desplazarse a una posición específica, o subir o bajar el nivel de zoom se usan los métodos `animateTo(GeoPoint)`, `zoomIn()` y `zoomOut()`.

e. Agregando otro botón para animar el zoom:

```
private Button btnAnimar = null;
      :
btnAnimar = (Button)findViewById(R.id.BtnAnimar);
      :
btnAnimar.setOnClickListener(new OnClickListener() {
      public void onClick(View arg0) {
      Double latitud = 37.40*1E6;
      Double longitud = -5.99*1E6;
      GeoPoint loc = new GeoPoint(latitud.intValue(), longitud.intValue());
      controlMapa.animateTo(loc);
      int zoomActual = mapa.getZoomLevel();
      for(int i=zoomActual; i<10; i++) controlMapa.zoomIn();
      }
});

Para desplazar el mapa un determinado número de pixeles, por ejemplo 40,  a otra
dirección se invoca a scrollBy():
private Button btnMover = null;
      :
btnMover = (Button)findViewById(R.id.BtnMover);
      :
btnMover.setOnClickListener(new OnClickListener() {
      public void onClick(View arg0) {
```

```
      controlMapa.scrollBy(40, 40);
      }
});
```

SECCIÓN III.

Uso de `Overlay`.

La información personalizada sobre un control `MapView` se realiza con capas `Overlay`.

a. Crear una clase java que herede de `Overlay` y sobrescribir el método `draw()`.

Se agrega un marcador sobre las coordenadas fijas. El método `draw()` recibe un objeto `Canvas` y sobre él se dibuja con los métodos `drawLine()`, `drawCircle()`, `drawText()` o `drawBitmap()`. Con la clase `Projection`, se realizan conversiones entre sistemas de referencia (pixeles y grados).

Crear un objeto `GeoPoint` que tome la latitud y longitud. Crear otro objeto `Projection`, con el método `getProjection()` de la clase `MapView` toma la posición actual sobre la que está centrada el mapa y el nivel de zoom para convertir la `latitud` y `longitud` en grados y las coordenadas `x` e `y` en pixeles, lo que se hace invocando al método `toPixels()`:

```
Double latitud = 37.40*1E6;
Double longitud = -5.99*1E6;
Projection projection = mapView.getProjection();
GeoPoint geoPoint = new GeoPoint(latitud.intValue(), longitud.intValue());
Point centro = new Point();
projection.toPixels(geoPoint, centro);
```

Para agregar un círculo o etiqueta sobre las coordenadas calculadas se incluye lo siguiente:

```
Paint p = new Paint();
p.setColor(Color.BLUE);
canvas.drawCircle(centro.x, centro.y, 5, p);
canvas.drawText("Sevilla", centro.x+10, centro.y+5, p);
```

El código completo debe ser similar al siguiente:

```
public class OverlayMapa extends Overlay {
      private Double latitud = 37.40*1E6;
      private Double longitud = -5.99*1E6;
      public void draw(Canvas canvas, MapView mapView, boolean shadow){
      Projection projection = mapView.getProjection();
      GeoPoint geoPoint = new GeoPoint(latitud.intValue(), longitud.intValue());
      if (shadow == false){
      Point centro = new Point();
      projection.toPixels(geoPoint, centro);
      Paint p = new Paint();
      p.setColor(Color.BLUE);
      canvas.drawCircle(centro.x, centro.y, 5, p);
      canvas.drawText("Sevilla", centro.x+10, centro.y+5, p);
      }
      }
}
```

Para añadir la capa al mapa se usa el método `onCreate()` y se obtiene la lista de capas con el método `getOverlays()`, se crea una nueva instancia de `OverlayMapa`, se agrega con el método `add()` y se redibuja el mapa con el método `postInvalidate()`:

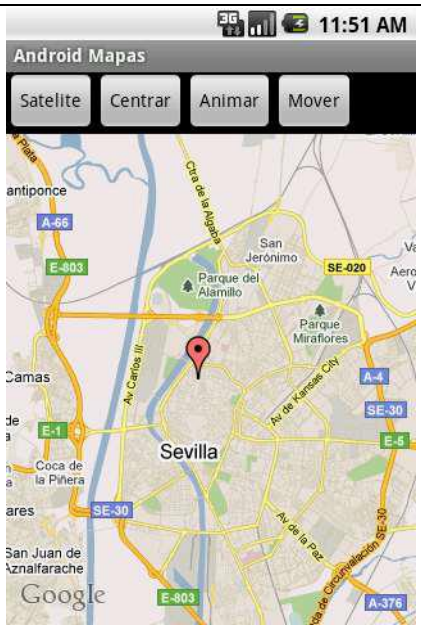
```
mapa = (MapView)findViewById(R.id.mapa);
      :
List<Overlay> capas = mapa.getOverlays();
OverlayMapa om = new OverlayMapa();
capas.add(om);
mapa.postInvalidate();
```

b. Probar la aplicación para mostrar el mapa centrado en las coordenadas y la información de la nueva capa:

Para incluir y dibujar un marcador gráfico con `drawBitmap()` sobre el mapa, se coloca una imagen del marcador en la carpeta /res/drawable:

```
Bitmap bm = BitmapFactory.decodeResource(
mapView.getResources(),
R.drawable.marcador_google_maps);
canvas.drawBitmap(bm, centro.x - bm.getWidth(), centro.y - bm.getHeight(), p);
```

c. Observar que se incluyó una imagen similar a una gota roja invertida, como se muestra enseguida:



SECCIÓN IV.

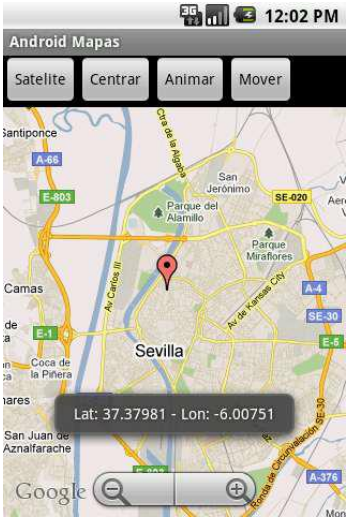
Eventos de usuario.

Para que el usuario pueda interactuar el sobre control se sobrescribe el método `onTap`() el cual proporciona las coordenadas de latitud y longitud que ha seleccionado el usuario.

a. Utilizar un `Toast` para mostrar las coordenadas seleccionadas:

```
public boolean onTap(GeoPoint point, MapView mapView){
      Context contexto = mapView.getContext();
      String msg = "Lat: " + point.getLatitudeE6()/1E6 + "-" + "Lon:
"+point.getLongitudeE6()/1E6;
      Toast toast = Toast.makeText(contexto, msg, Toast.LENGTH_SHORT);
      toast.show();
      return true; //regresa true si no hay digitación y no se notifica.
}
```

b. Ejecutar la aplicación para probar los nuevos cambios:


12:02 PM
Android Mapas
Satelite
Centrar
Animar
Mover
Sevilla
Camas
Coca de la Piñera
San Juan de Aznalfarache
Lat: 37.37981 - Lon: -6.00751
Google