|  |  |
| --- | --- |
|  | **UNIVERSIDAD DON BOSCO**  **FACULTAD DE ESTUDIOS TECNOLÓGICOS**  **COORDINACIÓN DE COMPUTACIÓN Y MÓVILES** |
| **Ciclo I** | **Desarrollo de Software para móviles** Guía de Laboratorio No. 6Mapas |

* **I Objetivos**
* Que el alumno conozca el uso de la API de Google Maps.
* Que el alumno personalice los mapas creados.
* **II Introducción**

En diciembre de 2012, Google presentaba la segunda versión de su API de Google Maps para Android. Esta nueva versión presenta muchas novedades interesantes, de las que cabe destacar las siguientes:

Integración con los Servicios de Google Play (Google Play Services) y la Consola de APIs.

Utilización a través de un nuevo tipo específico de fragment (MapFragment), una mejora muy esperada por muchos.

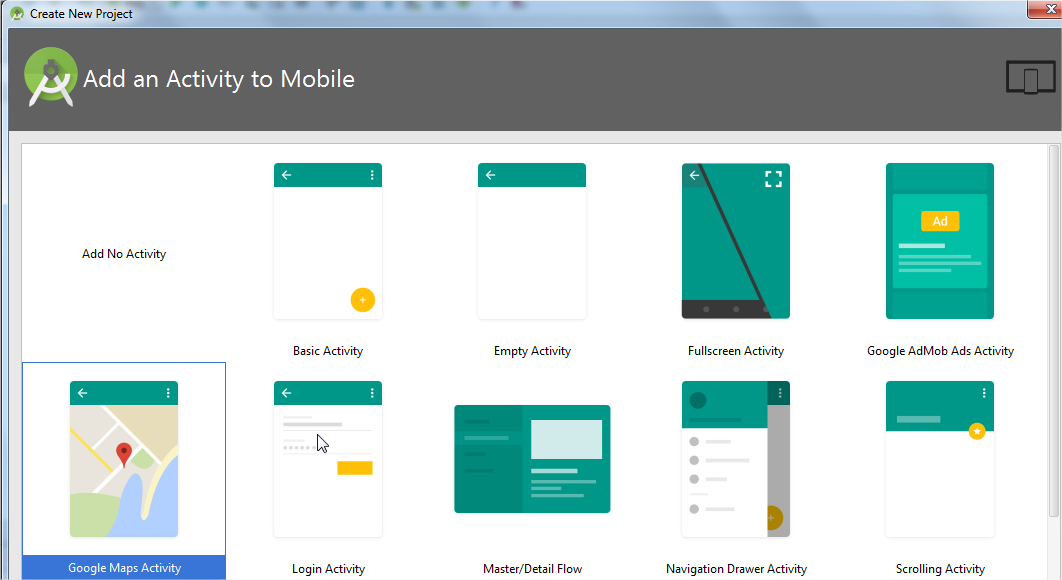
Utilización de mapas vectoriales, lo que repercute en una mayor velocidad de carga y una mayor eficiencia en cuanto a uso de ancho de banda.

Mejoras en el sistema de caché, lo que reducirá en gran medida las famosas áreas en blanco que tardan en cargar.

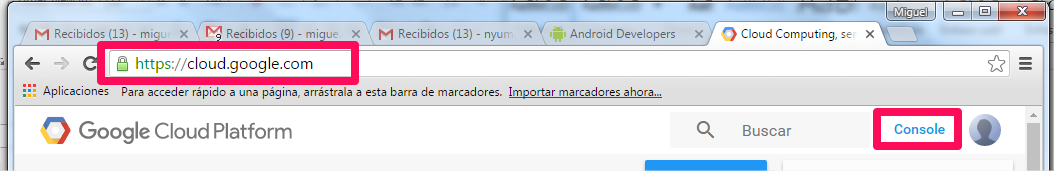
Los mapas son ahora 3D, es decir, podremos mover nuestro punto de vista de forma que lo veamos en perspectiva.

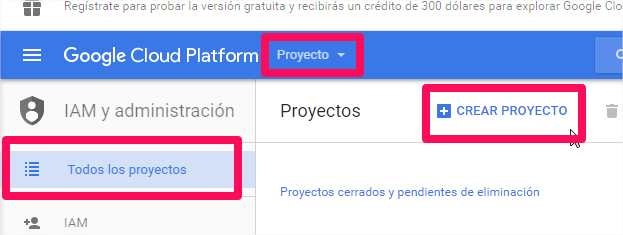
* **III Procendimiento**

Crear un proyecto nuevo denominado “Mapas”, seleccionar como actividad inicial “Google Maps Activity”

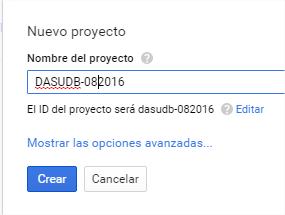


Desde la consola de Android Developer, dar de alta el servicio de Google Maps y obtener el ID. Nota: Los pasos que se muestran a continuación pueden cambiar próximamente, se le sugiere comprenda los pasos generales para entender qué es lo que se desea habilitar.

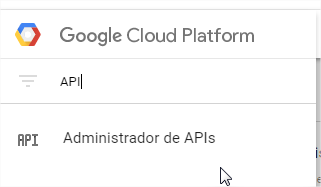


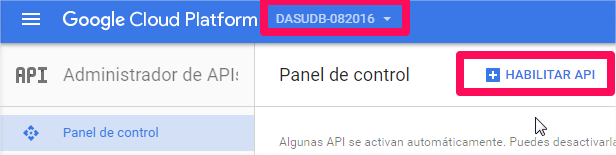


Establecer un nombre diferente al mostrado

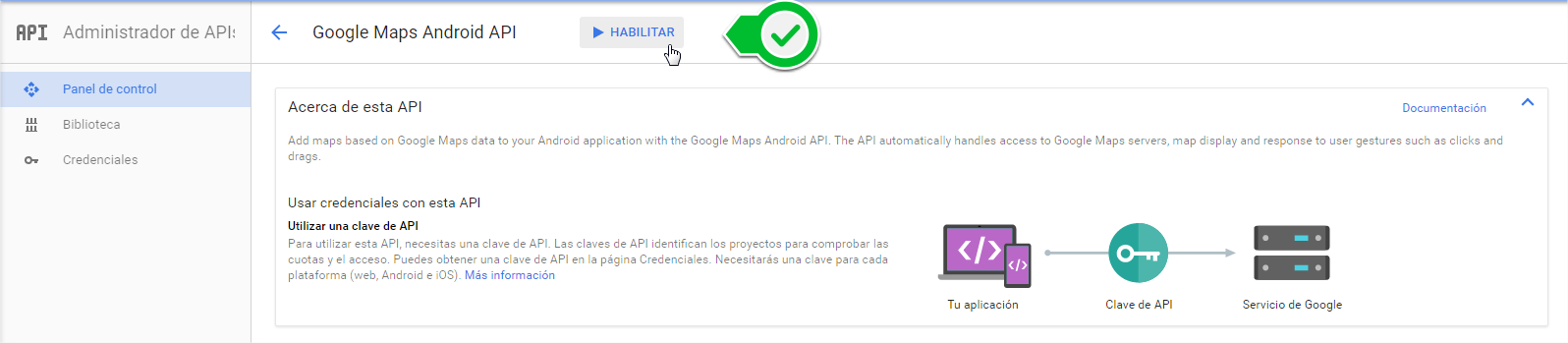


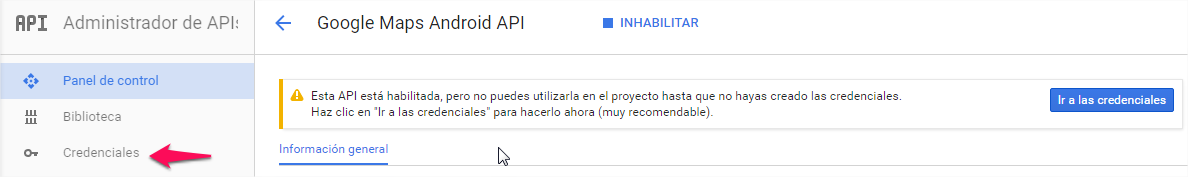
Buscar el proyecto creado y agregar las APIS que utilizaremos:



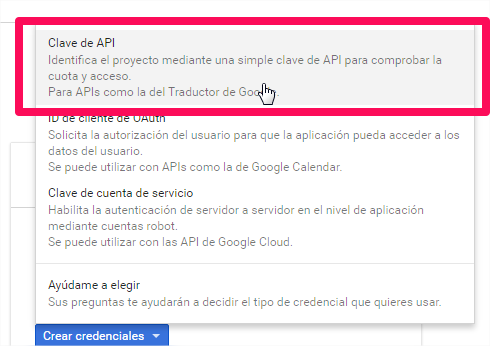
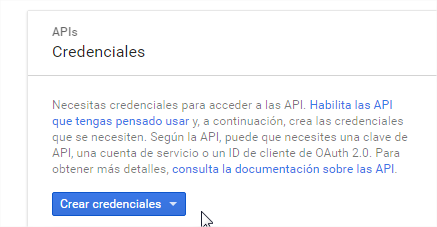


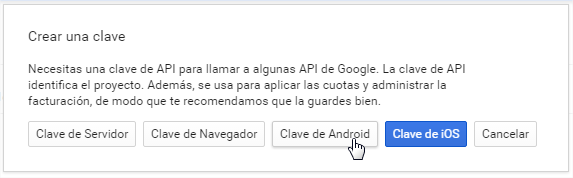






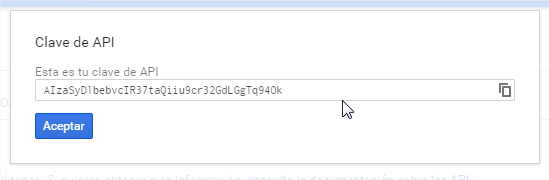
Crear credenciales:



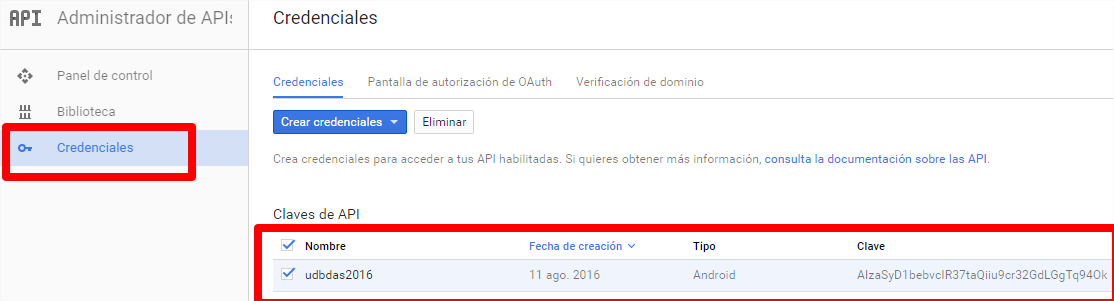


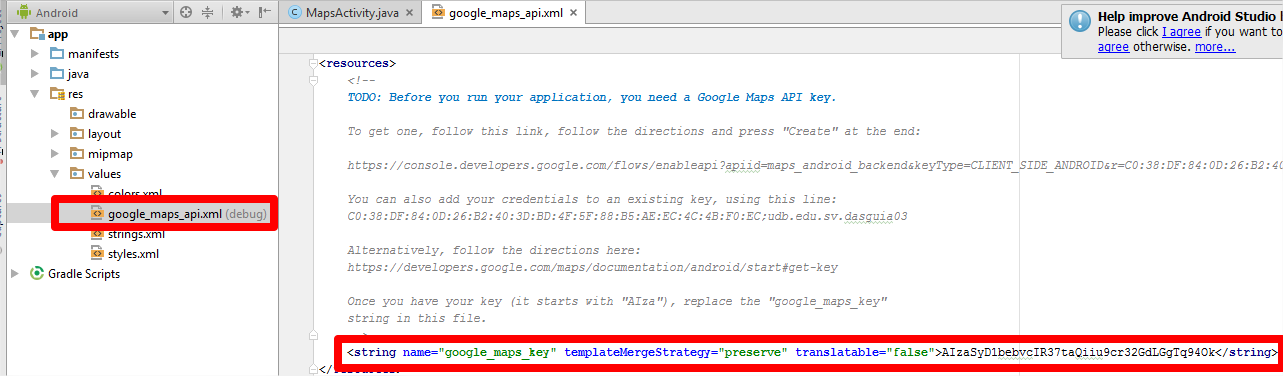


Obtendremos nuestra clave de API

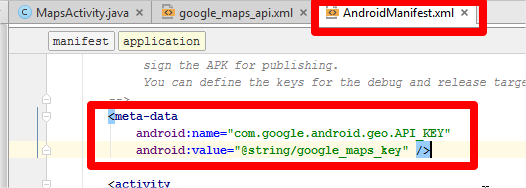


Éste será el valor de API que necesitamos utilizar en nuestro proyecto.





Ése valor es utilizado aquí:





Antes de ejecutar la aplicación:

Es necesario que usted utilice un dispositivo emulado que cuente con Google Play Services. Desde el sdk manager se encuentran diferentes rooms con tales características. Si utiliza genymotion, será necesario que realice la instalación manualmente de Google Play Services. Por otra parte, los dispositivos físicos, únicamente requieren tener el software actualizado.

**Agregaremos los permisos en nuestro AndroidManifest.xml:**



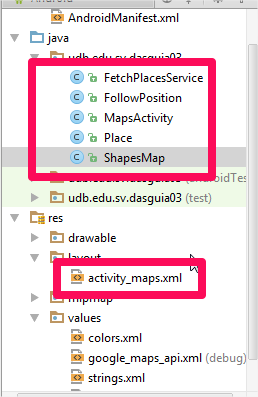
**Incluir el siguiente texto en strings.xml**

|  |
| --- |
| <**resources**>  <**string name="app\_name"**>DASGUIA03</**string**>  <**string name="title\_activity\_maps"**>Map</**string**>  <**string-array name="list\_map\_type"**>  <**item**>MAP\_TYPE\_NORMAL</**item**>  <**item**>MAP\_TYPE\_SATELLITE</**item**>  <**item**>MAP\_TYPE\_HYBRID</**item**>  </**string-array**> </**resources**> |

Agregar el archivo locationicon.png en la carpeta drawable

C:\Users\Miguel\Dropbox\UNIVERSIDAD 2016\CICLO II\DAS\Guias de laboratorio\icon.png

Al finalizar nuestro proyecto tendrá la siguiente estructura:



La clase **FetchPlacesService** es un servicio que se encargará de facilitar los datos de los marcadores. Por ahora los datos que obtenemos son estáticos, sin embargo, éstos serán dinámicos al obtenerse de un WebService en guías futuras

La clase **FollowPosition** hace uso de LocationManager para obtener la posición por medio de geolocalización.

La clase **Place** es un pojo para almacenar la estructura de los puntos.

La clase **ShapesMap** se encargará de dibujar en el lienzo figuras geométricas.

La clase **MapsActivity** es la clase por defecto creada en el proyecto. En ella utilizaremos todas las clases creadas previamente.

**Editar el Layout principal activity\_maps.xml**

|  |
| --- |
| <**LinearLayout  xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:map="http://schemas.android.com/apk/res-auto"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical"** >  <**Spinner  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:id="@+id/spinnerMapType"  android:entries="@array/list\_map\_type"**/>    <**SeekBar  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:id="@+id/seekBarZoom"  android:max="20"  android:progress="10"** />    <**fragment  android:id="@+id/map"  android:name="com.google.android.gms.maps.SupportMapFragment"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context="udb.edu.sv.dasguia03.MapsActivity"** />  </**LinearLayout**> |

**Crear la clase Java Place.java**

|  |
| --- |
| **package** udb.edu.sv.dasguia03;  **import** java.io.Serializable;  */\*\*  \* Clase POJO para objectos Place  \*/* **public class** Place **implements** Serializable{   *//Usar Refactor -> Encapsulate Fields para generar campos* **private** String **placeName**;  **private double lat**;  **private double lon**;   *//Utilizar Generate->Constructor* **public** Place(String placeName, **double** lat, **double** lon) {  **this**.**placeName** = placeName;  **this**.**lat** = lat;  **this**.**lon** = lon;  }     **public** String getPlaceName() {  **return placeName**;  }   **public void** setPlaceName(String placeName) {  **this**.**placeName** = placeName;  }   **public double** getLat() {  **return lat**;  }   **public void** setLat(**double** lat) {  **this**.**lat** = lat;  }   **public double** getLon() {  **return lon**;  }   **public void** setLon(**double** lon) {  **this**.**lon** = lon;  } } |

**Crear la clase FetchPlacesService**

|  |
| --- |
| **package** udb.edu.sv.dasguia03;  **import** android.app.IntentService; **import** android.app.Service; **import** android.content.Intent; **import** android.os.IBinder; **import** android.support.annotation.Nullable; **import** android.util.Log;  **import** com.google.android.gms.maps.model.LatLng;  **import** java.util.ArrayList; **import** java.util.List;  */\*\*  \* Created by Miguel on 11/08/2016.  \*/   //Implementa IntentService.  //Tome en cuenta que debe implementar el método onHandleIntent* **public class** FetchPlacesService **extends** IntentService {   *//Hará las veces de identificador* **public static** String *NOTIFICATION* = **"udb.edu.sv.dasguia03"**;   **public static** String *RESULT* = **"dataResult"**;   **private** ArrayList<Place> **result** = **new** ArrayList();    *//Un identificador de clase.  //se pasa como parámetro en súper.* **public** FetchPlacesService() {  **super**(**"fetchplaces"**);  }     *//Se ejecuta al iniciar el servicio y deja los datos preparados  // para ser tomados con BroadCastReceiver* @Override  **protected void** onHandleIntent(Intent intent) {   *//Llenaremos el ArrayList con nuestros datos.* **result**.add(**new** Place(**"Facultad de estudios Tecnológicos"**,  13.715578, -89.152609));  **result**.add(**new** Place(**"Edificio A"**,13.716021, -89.153399));  **result**.add(**new** Place(**"Edificio B"**,13.715769, -89.153387));  **result**.add(**new** Place(**"Facultad de estudios Tecnológicos"**,  13.715578, -89.152609));  **result**.add(**new** Place(**"Edificio A"**,13.716021, -89.153399));  **result**.add(**new** Place(**"Edificio B"**,13.715769, -89.153387));  **result**.add(**new** Place(**"Aula Magna A"**,13.715963, -89.153740));  **result**.add(**new** Place(**"Aula Magna B"**,13.715704, -89.153672));  **result**.add(**new** Place(**"Edificio R"**,13.716286, -89.153661));  **result**.add(**new** Place(**"Biblioteca"**,13.716837, -89.153570));  **result**.add(**new** Place(**"Colegio Don Bosco"**,13.716640, -89.150315));  **result**.add(**new** Place(**"Canchas"**,13.715568, -89.152015));  **result**.add(**new** Place(**"Edificio B"**,13.715769, -89.153387));   publishData();   }   *//Método custom que se encargará de publicar los datos  //para que sean capturados por MapsActivity* **public void** publishData(){   Intent intent = **new** Intent(*NOTIFICATION*);  intent.putExtra(*RESULT*, **result**);  sendBroadcast(intent);  } } |

**Crear la clase ShapesMap.java**

|  |
| --- |
| **package** udb.edu.sv.dasguia03;  **import** com.google.android.gms.maps.GoogleMap; **import** com.google.android.gms.maps.model.Circle; **import** com.google.android.gms.maps.model.CircleOptions; **import** com.google.android.gms.maps.model.LatLng; **import** com.google.android.gms.maps.model.Polygon; **import** com.google.android.gms.maps.model.PolygonOptions; **import** com.google.android.gms.maps.model.Polyline; **import** com.google.android.gms.maps.model.PolylineOptions;  **import** java.util.ArrayList;  */\*\*  \* Permite dibujar una serie de figuras geométricas en un mapa  \*/* **public class** ShapesMap {  GoogleMap **mMap**;   **public** ShapesMap(GoogleMap mMap){  **this**.**mMap** = mMap;  }    */\*\*  \*  \** ***@param points*** *\** ***@param width*** *\** ***@param color*** *\*/   /\*Dibujamos líneas a partir de la opción PolylineOptions.addAll\*/* **public void** drawLine(ArrayList<LatLng> points,**int** width, **int** color){  PolylineOptions options= **new** PolylineOptions();  options.addAll(points);  options.width(width);  options.color(color);  Polyline polyline = **mMap**.addPolyline(options);  polyline.setColor(color);  }   */\*\*  \* Dibujamos un polígono a partir de la opción PolygonOptions.addAll  \** ***@param points*** *\** ***@param strokeWidth*** *\** ***@param strokeColor*** *\** ***@param fillColor*** *\*/* **public void** drawPoligon(ArrayList<LatLng> points, **int** strokeWidth, **int** strokeColor, **int** fillColor){   PolygonOptions options = **new** PolygonOptions();  options.addAll(points);  Polygon polygon= **mMap**.addPolygon(options);  polygon.setStrokeColor(strokeColor);  polygon.setStrokeWidth(strokeWidth);  polygon.setFillColor(fillColor);  polygon.setGeodesic(**true**);    }    */\*\*  \* Dibujamos un círculo a partir de CircleOptions  \** ***@param point*** *\** ***@param radius*** *\** ***@param strokeColor*** *\** ***@param strokeWidth*** *\** ***@param fillColor*** *\*/* **public void** drawCircle(LatLng point, **int** radius, **int** strokeColor, **int** strokeWidth, **int** fillColor){   CircleOptions circleOptions = **new** CircleOptions();  circleOptions.center(point);  circleOptions.radius(radius);  circleOptions.strokeColor(strokeColor);  circleOptions.strokeWidth(strokeWidth);  circleOptions.fillColor(fillColor);   Circle circle = **mMap**.addCircle(circleOptions);  }   } |

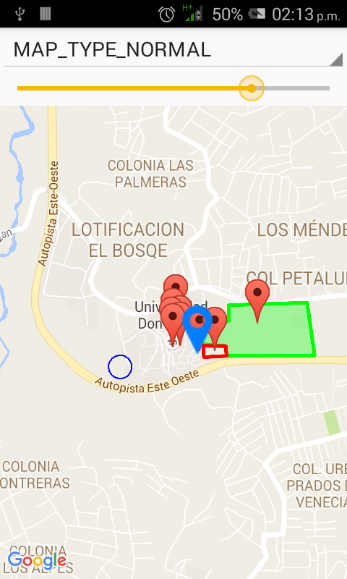
**Crear la clase FollowPosition.java**

|  |
| --- |
| **package** udb.edu.sv.dasguia03;  **import** android.Manifest; **import** android.content.Context; **import** android.content.pm.PackageManager; **import** android.location.Location; **import** android.location.LocationListener; **import** android.location.LocationManager; **import** android.os.Bundle; **import** android.support.v4.app.ActivityCompat;  **import** com.google.android.gms.maps.GoogleMap; **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;  */\*\*  \* Created by Miguel on 16/08/2016.  \*/* **public class** FollowPosition **implements** LocationListener {   **private** LocationManager **locationManager**;  Context **context**;   **private static int** *DISTANCE* = 1;  **private static int** *TIME* = 3000; *//3sec* **private** String **provider**;  Marker **point**;   **private** GoogleMap **mMap**;   **public** FollowPosition(GoogleMap mMap, Context context) {  **this**.setmMap(mMap);   **this**.**context** = context;  */\*\*\*\*\*\*\*\*\*\* Asigna Gps location service LocationManager object \*\*\*\*\*\*\*\*\*\*\*/* **locationManager** = (LocationManager) context.getSystemService(Context.***LOCATION\_SERVICE***);   **if**(**locationManager**.isProviderEnabled(LocationManager.***GPS\_PROVIDER***)){  **provider** = LocationManager.***GPS\_PROVIDER***;  }**else**{  **provider** = LocationManager.***NETWORK\_PROVIDER***;  }   }   **public void** register(Context context) {   *//NOS MOSTRARA UN ERROR . SELECCIONE LAS OPCIONES OFRECIDAS POR EL IDE  //PARA SOLVERTAR EL PROBLEMA* **if** (ActivityCompat.*checkSelfPermission*(context, Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(context, Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  *//* ***TODO: Consider calling*** *// ActivityCompat#requestPermissions  // here to request the missing permissions, and then overriding  // public void onRequestPermissionsResult(int requestCode, String[] permissions,  // int[] grantResults)  // to handle the case where the user grants the permission. See the documentation  // for ActivityCompat#requestPermissions for more details.* **return**;  }     *//INICIALIZAMOS LA LOCALIZACIÓN DEL DISPOSITIVO  //RECUERDE LAS OPCIONES:  //\*PROVEEDOR  //TIEMPO DE ACTUALIZACION  //DISTANCIA MINIMA  //LISTENER (EN ESTE CASO EL LISTENER ES LA MISMA CLASE  //CONSULTE AL INSTRUCTOR PARA MAS DETALLES* **this**.**locationManager**.requestLocationUpdates(**provider**,  *TIME*, *// 30 segundos* 10, **this**); *// 10 metros* }   **public void** unRegister(Context context) {   **if** (ActivityCompat.*checkSelfPermission*(context, Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED*** && ActivityCompat.*checkSelfPermission*(context, Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {  *//* ***TODO: Consider calling*** *// ActivityCompat#requestPermissions  // here to request the missing permissions, and then overriding  // public void onRequestPermissionsResult(int requestCode, String[] permissions,  // int[] grantResults)  // to handle the case where the user grants the permission. See the documentation  // for ActivityCompat#requestPermissions for more details.* **return**;  }   **this**.**locationManager**.removeUpdates(**this**);  }     @Override  **public void** onLocationChanged(Location location) {   LatLng latLng = **new** LatLng(location.getLatitude(), location.getLongitude());   MarkerOptions currentPosition = **new** MarkerOptions();  currentPosition.position(latLng);  currentPosition.title(**"Aquí estoy"**);  currentPosition.icon(BitmapDescriptorFactory.*fromResource*(R.drawable.***locationicon***));   *//Quitamos el viejo punto* **if**(**point** != **null**) **point**.remove();   *//Agregamos el nuevo punto* **point** = getmMap().addMarker(currentPosition);   }     @Override  **public void** onStatusChanged(String provider, **int** status, Bundle extras) {   }   @Override  **public void** onProviderEnabled(String provider) {   }   @Override  **public void** onProviderDisabled(String provider) {   }   **public** GoogleMap getmMap() {  **return mMap**;  }   **public void** setmMap(GoogleMap mMap) {  **this**.**mMap** = mMap;  } } |

**Editar la clase MapsActivitivy.java creada por defecto**

|  |
| --- |
| **package** udb.edu.sv.dasguia03;  **import** android.content.BroadcastReceiver; **import** android.content.Context; **import** android.content.Intent; **import** android.content.IntentFilter; **import** android.graphics.Color; **import** android.support.v4.app.FragmentActivity; **import** android.os.Bundle; **import** android.view.View; **import** android.widget.AdapterView; **import** android.widget.SeekBar; **import** android.widget.Spinner; **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.CameraPosition; **import** com.google.android.gms.maps.model.LatLng; **import** com.google.android.gms.maps.model.MarkerOptions;  **import** java.util.ArrayList;  **public class** MapsActivity **extends** FragmentActivity **implements** OnMapReadyCallback {   **private** GoogleMap **mMap**;   ArrayList<Place> **places**;   Spinner **spinnerMapType**;  SeekBar **seekBarZoom**;  LatLng **defaultLatLng** = **new** LatLng(13.714966, -89.155755);   FollowPosition **followPosition**;   @Override  **protected void** onCreate(Bundle savedInstanceState) {  **super**.onCreate(savedInstanceState);  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**);    **seekBarZoom** = (SeekBar) findViewById(R.id.***seekBarZoom***);   *//HAGA USO DEL ASISTENTE PARA CREAR setOnSeekBarChangeListener  //El único método que modificará es onProgressChanged* **seekBarZoom**.setOnSeekBarChangeListener(**new** SeekBar.OnSeekBarChangeListener() {  @Override  **public void** onProgressChanged(SeekBar seekBar, **int** progress, **boolean** fromUser) {   *//CODIGO INTERIOR CREADO POR USTED* chooseMoveCamera(**mMap**, **defaultLatLng**, progress);   }   @Override  **public void** onStartTrackingTouch(SeekBar seekBar) {   }   @Override  **public void** onStopTrackingTouch(SeekBar seekBar) {   }  });   **spinnerMapType** = (Spinner) findViewById(R.id.***spinnerMapType***);   *//HAGA USO DEL ASISTENTE PARA CREAR setOnItemSelectedListener* **spinnerMapType**.setOnItemSelectedListener(**new** AdapterView.OnItemSelectedListener() {  @Override  **public void** onItemSelected(AdapterView<?> parent, View view, **int** position, **long** id) {   *//CODIGO INTERIOR CREADO POR USTED* String mapType = **spinnerMapType**.getSelectedItem().toString();  **if** (**mMap** == **null**) **return**;   **if** (mapType.equals(**"MAP\_TYPE\_NORMAL"**)) {  **mMap**.setMapType(GoogleMap.***MAP\_TYPE\_NORMAL***);  } **else if** (mapType.equals(**"MAP\_TYPE\_SATELLITE"**)) {  **mMap**.setMapType(GoogleMap.***MAP\_TYPE\_SATELLITE***);  } **else if** (mapType.equals(**"MAP\_TYPE\_HYBRID"**)) {  **mMap**.setMapType(GoogleMap.***MAP\_TYPE\_HYBRID***);  }  *//FIN DE CODIGO INTERIOR CREADO POR USTED* }   @Override  **public void** onNothingSelected(AdapterView<?> parent) {   }  });      }    *//Broadcast Receiver.  //Permanecerá escuchando por actualizaciones de FetchPlacesService  // (Servicio que intentará descargar los datos)  //HAGA USO DEL ASISTENTE PARA CREAR BroadcastReceiver* **private** BroadcastReceiver **broadcastReceiver** = **new** BroadcastReceiver() {  @Override  **public void** onReceive(Context context, Intent intent) {    *//CÓDIGO INTERIOR CREADO POR USTED* Bundle bundle = intent.getExtras();  **if** (bundle != **null**) {  **places** = (ArrayList<Place>) bundle.getSerializable(FetchPlacesService.*RESULT*);  **if** (**places** != **null** && **places**.size() > 0) {  **if** (**mMap** != **null**) {  **for** (Place tmp : **places**) {  LatLng tmpLatLng =  **new** LatLng(tmp.getLat(), tmp.getLon());  **mMap**.addMarker(**new** MarkerOptions().  position(tmpLatLng).  title(tmp.getPlaceName())  );  }  }  }  }     *//FIN DEL CÓDIGO INTERIOR CREADO POR USTED* }  };   @Override  **protected void** onResume() {  **super**.onResume();  registerReceiver(**broadcastReceiver**, **new** IntentFilter(FetchPlacesService.*NOTIFICATION*));  */\*\*/* Intent intent = **new** Intent(**this**, FetchPlacesService.**class**);   startService(intent);   **if** (**followPosition** != **null**) {  **followPosition**.register(MapsActivity.**this**);   }   }   @Override  **protected void** onPause() {  unregisterReceiver(**broadcastReceiver**);  **if** (**followPosition** != **null**)  **followPosition**.unRegister(MapsActivity.**this**);  **super**.onPause();  }    */\*\*  \* 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;   **followPosition** = **new** FollowPosition(**this**.**mMap**, MapsActivity.**this**);   **followPosition**.register(MapsActivity.**this**);   *//Moveremos la cámara a la Universidad Don Bosco* **mMap**.moveCamera(CameraUpdateFactory.*newLatLng*(**defaultLatLng**));  chooseMoveCamera(**mMap**, **defaultLatLng**, 10);   drawShapes();   }    *//El siguiente método permitirá movernos de manera animada  // a una posición del mapa* **private void** chooseMoveCamera(GoogleMap googleMap, LatLng tmpLatLng, **int** zoom){  CameraPosition cameraPosition =  **new** CameraPosition.Builder().zoom(zoom).target(tmpLatLng).build();   googleMap.animateCamera  (CameraUpdateFactory.*newCameraPosition*(cameraPosition));  }   *//El siguiente método custom permite agregar diferentes figuras* **private void** drawShapes(){   ShapesMap shapesMap = **new** ShapesMap(**this**.**mMap**);   *//PolyLines* ArrayList<LatLng> lines = **new** ArrayList<>();   lines.add(**new** LatLng (13.715777, -89.152472) );  lines.add(**new** LatLng (13.715342, -89.152437) );  lines.add(**new** LatLng (13.715389, -89.151542) );  lines.add(**new** LatLng (13.715768, -89.151579) );  lines.add(**new** LatLng (13.715777, -89.152472) );   *//Llamado al método custom drawLine de shapesMap* shapesMap.drawLine(lines,5, Color.***RED***);   ArrayList<LatLng> linesD = **new** ArrayList<>();    ArrayList<LatLng> poligon = **new** ArrayList<>();  poligon.add(**new** LatLng(13.715389, -89.151542));  poligon.add(**new** LatLng(13.715407, -89.148059));  poligon.add(**new** LatLng(13.717434, -89.148355));  poligon.add(**new** LatLng(13.717336, -89.151456));  poligon.add(**new** LatLng(13.716741, -89.151447));  poligon.add(**new** LatLng(13.716692, -89.152468));  poligon.add(**new** LatLng(13.715841, -89.152558));  poligon.add(**new** LatLng(13.715768, -89.151579));  poligon.add(**new** LatLng(13.715389, -89.151542));    *//Transparencia  //Valor Hexadecimal, transparencia + color  //0x: Valor hexadecimal  //2F: Trasparencia  //00FF00: Color Hexadecimal* shapesMap.drawPoligon(poligon,5,Color.***GREEN***,0x2F00FF00);     *//Agregando Circulo* LatLng circlePoint = **new** LatLng(13.714966, -89.155755);  shapesMap.drawCircle(circlePoint,50,Color.***BLUE***,2,Color.***TRANSPARENT***);   } } |

**Resultado esperado:**



**IV. Investigación complementaria.**

Realizar una aplicación que incorpore mapas para indicar los parques de su comunidad.

Utilizar figuras geométricas para indicar las zonas de riegos naturales (deslaves, cárcavas, ríos, etc) de su comunidad.

Investigue cómo agregar a un marcador el “MarkerClickListener” para mostrar en una actividad nueva, imágenes del sitio o punto seleccionado.

**V. Bibliografía**

<http://www.sgoliver.net/blog/mapas-en-android-google-maps-android-api-v2-i/>

<http://www.sgoliver.net/blog/mapas-en-android-google-maps-android-api-v2-ii/>

<http://www.sgoliver.net/blog/mapas-en-android-google-maps-android-api-v2-iii/>

Hoja de cotejo.

Docente:

Máquina No:

GL:

a

**HOJA DE EVALUACIÓN**

Alumno:

No.:

Fecha:

Carnet:

Docente:

Título de la guía:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Actividad a evaluar** | **Criterio a evaluar** | **Cumplió** | | **Puntaje** |
| **SI** | **NO** |
| Desarrollo | Realizó los ejemplos de guía de práctica (40%) |  |  |  |
| Presentó todos los problemas resueltos (20%) |  |  |  |
| Funcionan todos correctamente y sin errores (30%) |  |  |  |
| Envió la carpeta comprimida y organizada adecuadamente en subcarpetas de acuerdo al tipo de recurso (10%) |  |  |  |
| PROMEDIO: |  | | |
| Investigación complementaria | Envió la investigación complementaria en la fecha indicada (20%) |  |  |  |
| Resolvió todos los ejercicios planteados en la investigación (40%) |  |  |  |
| Funcionaron correctamente y sin ningún mensaje de error a nivel de consola o ejecución (40%) |  |  |  |
| PROMEDIO: |  |  |  |
|  |  | | |