**Asignatura**: Desarrollo de Aplicaciones Móviles Nativas.

**Tema**. Mapas 2. El uso de mapas. Parte 2.

# 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()

* 1. 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);



**Figura 1**. Probando el botón se muestra la vista de satélite:

* 1. 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.

* 1. 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);

}

});



**Figura 2**. 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().

* 1. 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);

}

});

# Uso de Overlay.

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

* 1. 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();

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



**Figura 3**. 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);

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



**Figura 4**. Imagen similar a una gota roja invertida.

# 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.

* 1. 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.

}

* 1. Ejecutar la aplicación para probar los nuevos cambios:



**Figura 5**. El mapa con los cambios.

**NOTA**: Capturar las imágenes, de la ejecución de los ejercicios en un documento y guardarlo con la sintaxis

AlumnoMapas2Grupo.pdf. Enviar el archivo al sitio indicado por el profesor.