



Mapas

OpenStreetMap

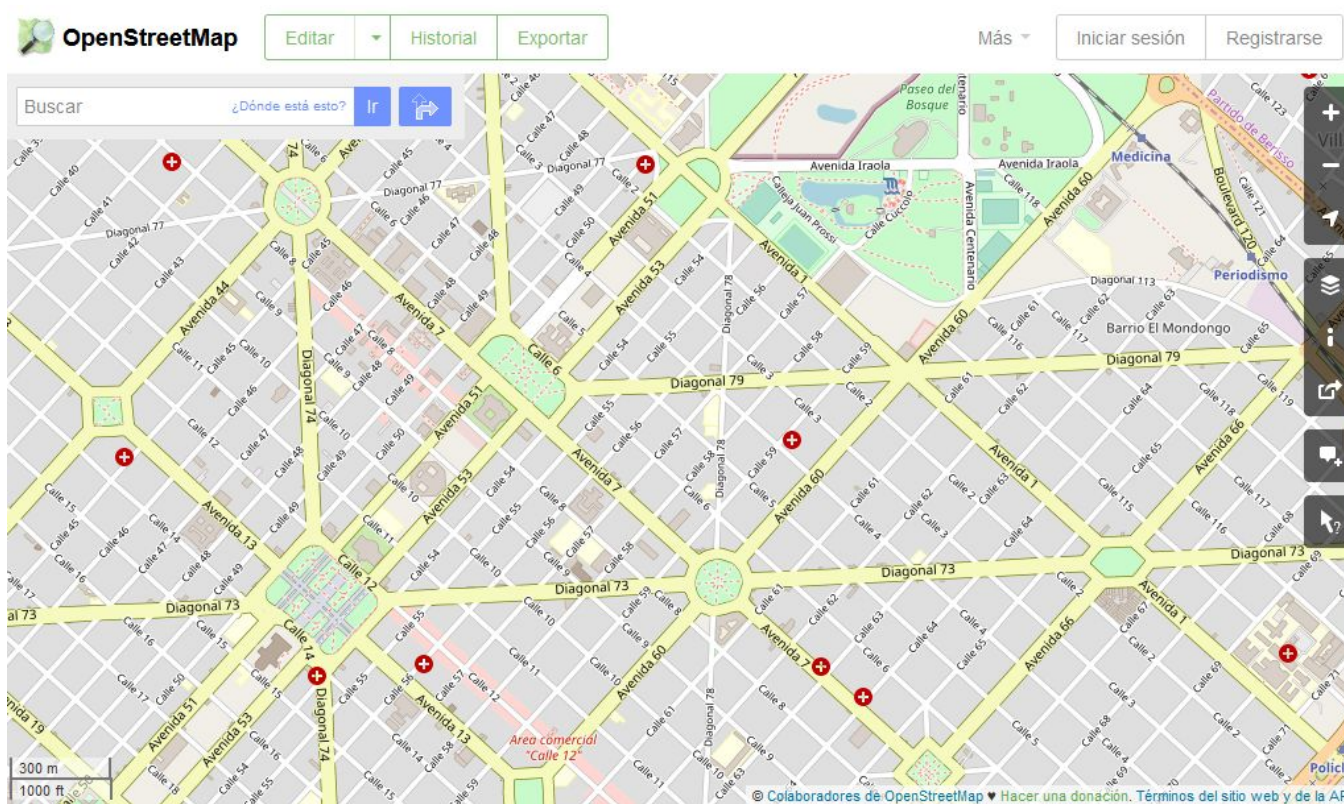


- **OpenStreetMap (OSM)** es un proyecto que posibilita la utilización de **mapas y datos geográficos** en forma **libre** sin correr riesgo de infringir restricciones legales.
- No solo se encarga del **dibujado de los mapas**, sino que también tiene funciones de **trazado de rutas**, **geo-codificación** (encontrar coordenadas de un objeto) y **análisis espacial**.
- Permite que los **usuarios** puedan **crear sus propios mapas**.

OpenStreetMap



Los mapas **OSM** contienen formas que definen las **zonas urbanas** (edificios, calles, plazas, etc.), **señales de tráfico** como semáforos, indicadores de **lugares importantes** (cafeterías, farmacias, iglesias, etc.), y **restricciones** como lo son las flechas que indican el sentido de las calles.



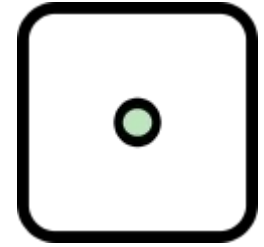
OpenStreetMap Mapa



- Los mapas **OSM** se organizan a partir de tres componentes principales que son el núcleo del modelo de datos: los **nodos**, las **vías** y las **relaciones**.
- Tomando como base estos **tres elementos** se organiza toda la información del mapa **OSM** y las **nuevas propiedades** (u otros detalles) que se añaden al mismo se construyen a partir de alguno de estos tres elementos.
- Otra característica fundamental son las **etiquetas**, que se utilizan para describir **propiedades que poseen los elementos en el mapa**.

OpenStreetMap

Nodos



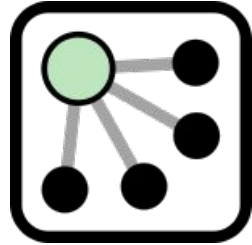
- Un **nodo** es un elemento puntual que se define por su **posición** (**latitud**, **longitud** y si corresponde una **altitud**), un **identificador** (ID) y una **etiqueta** que se añade para especificar propiedades tales como el número de pisos en edificios, calles de distintas alturas, cruces o puentes.
- Los **nodos** se usan principalmente en **agrupamientos** con el fin de definir elementos de mayor complejidad, como por ejemplo las **vías**.
- Así mismo se pueden utilizar para definir características puntuales en el mapa.
- El **identificador** de un nodo **es único**. Cuando un nodo se elimina, su ID no vuelve a ser utilizado.

OpenStreetMap

Vías

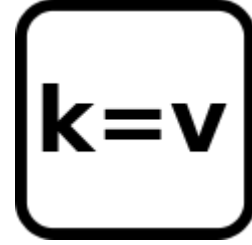


- Una **vía** es una lista ordenada de nodos definida por etiquetas y sirven para determinar rutas, caminos, edificios, ríos, etc.
- Cada vía tiene que estar conformada por **al menos dos nodos** y como máximo dos mil nodos.
- Al igual que los nodos, las vías tienen un **identificador único**.
- Las vías pueden ser cerradas o abiertas. Las vías cerradas son áreas que delimitan una superficie, como por ejemplo las plazas.



OpenStreetMap

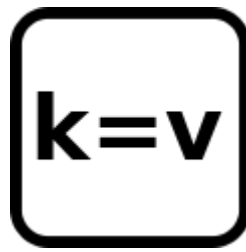
Relaciones y Etiquetas



- Las **relaciones** son **listas ordenadas de nodos**, **vías** e incluso de otras **relaciones** que definen **vinculaciones físicas o lógicas** entre todos los **componentes** de las listas. Un ejemplo de su uso es la definición de los **recorridos de los autobuses**, **fronteras**, **bosques**, **lagos**, etc.
- Las **etiquetas** describen **características de los elementos** del mapa. Constan de **dos campos**, uno denominado **key** que sirve como **definición de la característica** y otro llamado **value** que especifica el **valor de dicha característica**.

OpenStreetMap

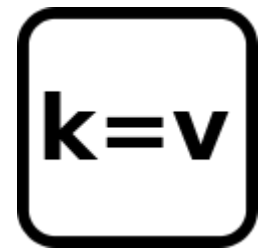
Etiquetas más comunes



- A continuación se detallan las **etiquetas más comunes** que podemos encontrar en un mapa **OSM**:
 - Para definir el tipo de una carretera se utiliza **highway**, **busway** o **cycleway**. La primera es la más utilizada y sus valores pueden ser *Primary* (ruta principal), *Secondary* (rutas secundarias), *Residential* (para accesos a zonas residenciales o urbanizaciones), *Pedestrian* (zonas peatonales), *Road* (cuando no está claro su clasificación), *Construction* (para vías en proceso de construcción o en obras), entre otros.
 - La etiqueta **route** define características tales como nombre, origen y destino o con una función determinada como por ejemplo las rutas de transporte público.

OpenStreetMap

Etiquetas más comunes



- Para definir el **número de carriles** se utiliza *lanes*. Así mismo, a los carriles se les puede asignar **restricciones** para la conducción como por ejemplo la velocidad máxima.
- La etiqueta *emergency* define un elemento relacionado con una situación de emergencia como una **estación de policías**, una **estación de bomberos**, etc.
- Los edificios se marcan con la etiqueta *building* y junto a esta se utiliza la etiqueta *addr* para definir la dirección del mismo, suele ir seguida por *addr:street* para la calle o *addr:full* si se define todo en el mismo campo.
- Entre muchas otras....

OSMDroid

- El proyecto **OSMDroid** se desarrolló para crear una **alternativa gratuita** a la clase **MapView** (API 1) **de Android**. Esta clase tan solo se puede usar con los mapas de Google, al igual que solo puede hacer peticiones de rutas al servidor de Google, el cual no ofrece este servicio de forma gratuita ni libre.
- **OSMDroid** también nos permite trabajar con **mapas offline**, es decir, en lugar de ir descargando los mapas mientras se navega, se pueden recuperar de una copia almacenada en el dispositivo - hay que tener cuidado dado que algunos proveedores de mapas lo consideran como una infracción de *copyright*-.

OSMDroid

OSMBonusPack

- **OSMBonusPack** es una **librería** que se presenta como una **extensión** de **OSMDroid** y utiliza muchas de las clases de la librería anterior por lo que **no se puede utilizar OSMBonusPack sin tener una versión instalada y actualizada de la OSMDroid**.
- Las **características** más importantes de **OSMBonusPack** son que **incorpora clases y métodos para el trazado de rutas**, junto con una **mejora** muy importante en el **sistema de marcadores**.

OSMDroid

MapView

- **MapView** es la **clase** encargada de crear la **vista del mapa** y permite principalmente **seleccionar el proveedor de mapas** deseado, con el método ***setTileSource()*** , y **obtener las capas u *overlays*** del mapa (polilíneas dibujadas, marcadores, *listeners* de eventos, etc).
- También posee un controlador llamado **MapController** para realizar operaciones tales como: **hacer zoom, centrar el mapa en una coordenada** específica, **desplazar el mapa hacia una posición** específica, etc.
- La **implementación incluida en la librería** de esta clase **modifica el *listener* de deslizamiento con el dedo** sobre el mapa para reemplazarlo con uno propio que realiza la operación en la dirección opuesta. También se **modifica la semántica del doble *click***, ahora cuando este evento es detectado se realiza una ampliación sobre el mapa.

OSMDroid

Marker y GeoPoint

- **Marker** es una **clase** que permite **crear** y **situar marcadores** en el mapa.
- **GeoPoint** es una **clase** que se compone por una **latitud** y una **longitud**. Representa las **coordenadas reales** sobre el mapa, por lo que es utilizado al momento de **desplazarse** hacia lugares concretos, **situar marcadores** u otros elementos en el mapa.

OSMDroid

MyLocationNewOverlay

- **MyLocationNewOverlay** es la **clase** encargada de obtener la **ubicación actual del usuario** y su **representación sobre el mapa**.
- Si el **dispositivo** posee un mecanismo para obtener su orientación, **brújula interna**, aparecerá una **flecha** en la dirección en que se halle el dispositivo respecto del mapa. En caso contrario simplemente aparecerá el **dibujo de una persona** para señalar la posición.
- Lleva a cabo el **cambio de orientación del mapa** y permite **activar o desactivar** el **seguimiento automático** de nuestra posición.
- Además permite **informar** sobre: la **velocidad de desplazamiento** y la **precisión** del método de localización.

OSMDroid

Instalación y Permisos

Al usar **Gradle** con **OSMDroid** se debe incorporar la dependencia *implementation 'org.osmdroid:osmdroid-android:6.1.0'* en el archivo **build.gradle**.

En el **AndroidManifest.xml** se deben dar las siguiente autorizaciones:

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


OSMDroid

Permisos (cont.)

En el **Activity** se deben solicitar las siguiente autorizaciones:

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=
    PERMISSION_GRANTED) {
    val permiso = arrayOf<String>(Manifest.permission.ACCESS_FINE_LOCATION)
    ActivityCompat.requestPermissions(this, permiso, 1)
}

override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>,
    grantResults: IntArray){
    super.onRequestPermissionsResult(requestCode, permissions!!, grantResults)
    if (requestCode == 1) {
        if (grantResults.size > 0 && grantResults[0] == PERMISSION_GRANTED)
            Toast.makeText(this@MainActivity, "Permission Granted", Toast.LENGTH_SHORT).show()
        else
            Toast.makeText(this@MainActivity, "Permission Denied", Toast.LENGTH_SHORT).show()
    }
}
```

OSMDroid Layout

También se debe crear un *layout* como el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <org.osmdroid.views.MapView android:id="@+id/mapview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tilesource="Mapnik"/>
</RelativeLayout>
```

Si se utiliza Android Studio como IDE, probablemente ya haya creado un layout por defecto, el cual normalmente es llamado: "src/main/res/layouts/activity_main.xml".

OSMDroid

Main Activity

Crear un *main activity* como la siguiente:

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import org.osmdroid.tileprovider.tilesource.TileSourceFactory
import org.osmdroid.views.MapView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val mapView = findViewById(R.id.mapview) as MapView
        mapView.setTileSource(TileSourceFactory.MAPNIK)
        mapView.setUseDataConnection(true)
        mapView.isClickable = true
    }
}
```


OSMDroid

Main Activity

Crear un *main activity* como la siguiente:

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import org.osmdroid.tileprovider.tilesource.TileSourceFactory
import org.osmdroid.views.MapView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val mapView = findViewById(R.id.mapview) as MapView
        mapView.setTileSource(TileSourceFactory.MAPNIK)
        mapView.setUseDataConnection(true)
        mapView.isClickable = true
    }
}
```



Una vez que
todo está
completo, la
app en
ejecución se
ve...

OSMDroid

Main Activity

Crear un *main activity* como la siguiente:

```
import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import org.osmdroid.tileprovider.tilesource.TileSourceFactory
import org.osmdroid.views.MapView

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        val mapView = findViewById(R.id.mapview) as MapView
        mapView.setTileSource(TileSourceFactory.MAPNIK)
        mapView.setUseDataConnection(true)
        mapView.isClickable = true
    }
}
```

Una vez que
todo está
completo, la
app en
ejecución se
ve...



OSMDroid

Zoom y default view point

Para **incorporar la función de zoom** con dos dedos (*multi-touch*), agregamos: `map.setMultiTouchControls(true)`

Si se desea **mover el mapa a una posición** por defecto y con un **zoom determinado**, se debe acceder al **MapController** como se muestra en siguiente segmento de código:

```
val mapViewController = mapView.controller
mapViewController.setZoom(16.5)
val Somewhere = GeoPoint(-34.9193, -57.9547)
mapViewController.setCenter(Somewhere)
```

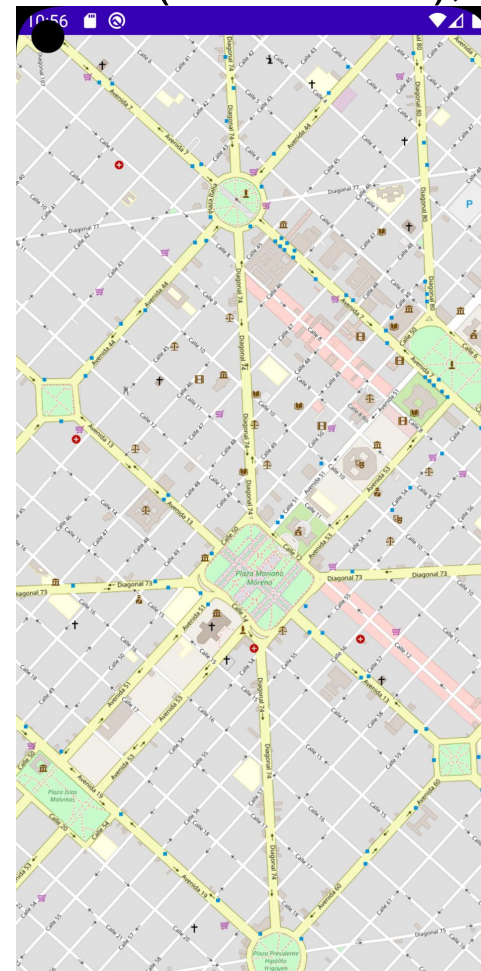
OSMDroid

Zoom y default view point

Para incorporar la función de **zoom** con dos dedos (*multi-touch*), agregamos: `map.setMultiTouchControls(true);`

Si se desea **mover el mapa a una posición** por defecto y con un **zoom determinado**, se debe acceder al **MapController** como se muestra en siguiente segmento de código:

```
val mapViewController = mapView.controller  
mapViewController.setZoom(16.5)  
val Somewhere = GeoPoint(-34.9193, -57.9547)  
mapViewController.setCenter(Somewhere)
```



OSMDroid

Mylocation

Para **marcar la posición actual** del dispositivo y **centrar el mapa** en ella, debemos utilizar la clase **MyLocationNewOverlay** y mediante el método ***enableMyLocation()*** moveremos el mapa a nuestra posición actual..

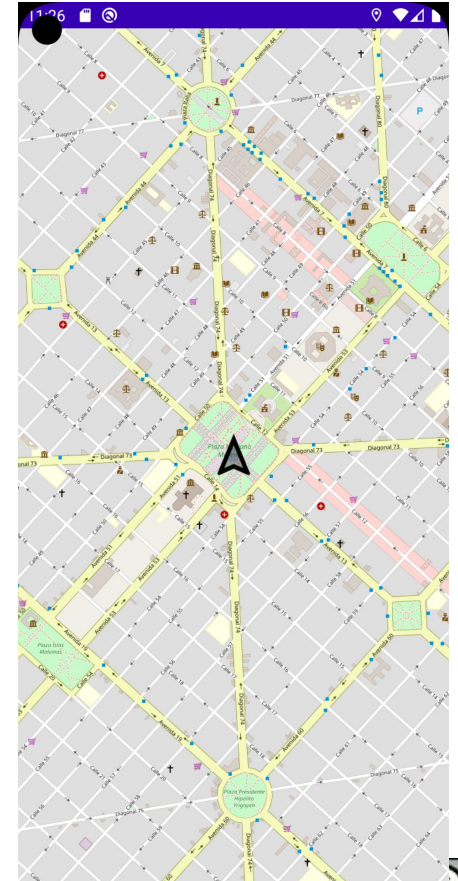
```
var myLocationOverlay = MyLocationNewOverlay(mapView)
myLocationOverlay.enableMyLocation()
```

OSMDroid

Mylocation

Para **marcar la posición actual** del dispositivo y **centrar el mapa** en ella, debemos utilizar la clase **MyLocationNewOverlay** y mediante el método ***enableMyLocation()*** moveremos el mapa a nuestra posición actual..

```
var myLocationOverlay = MyLocationNewOverlay(mapView)
mapView.overlays.add(myLocationOverlay)
myLocationOverlay.enableMyLocation()
```



OSMDroid

Markers

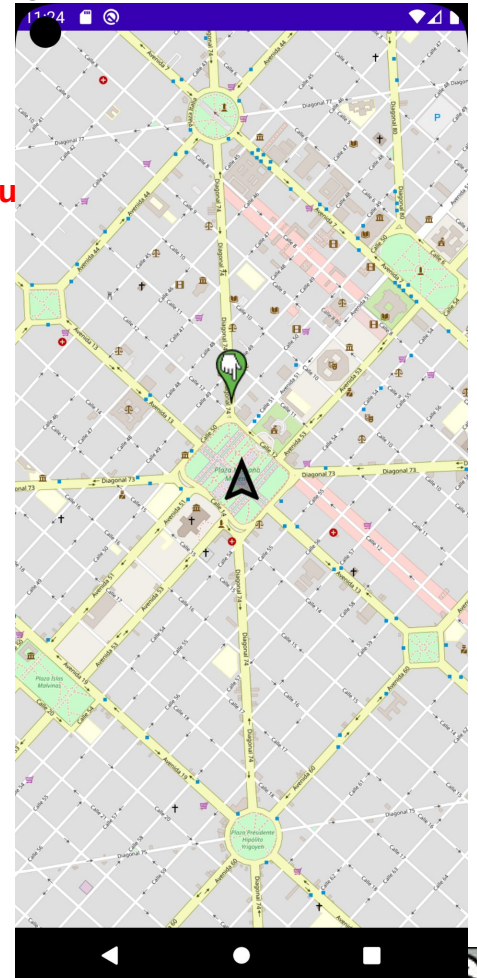
Para **añadir marcas** en puntos específicos se debe **definir** una **lista de puntos** con una **serie de atributos** particulares y **crear una nueva capa** que contenga esta información.

```
val poiIcon = ResourcesCompat.getDrawable(resources, R.drawable.marker_default, null)
val poiMarker = Marker(mapView)
poiMarker.title = "La Plata."
poiMarker.snippet = "Ciudad de La Plata "
poiMarker.position = Somewhere
poiMarker.icon = poiIcon
poiMarker.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM)
mapView.overlays.add(poiMarker)
```

OSMDroid Markers

Para **añadir marcas** en puntos específicos se debe **definir** una **lista de puntos** con una **serie de atributos** particulares y **crear una nueva capa** que contenga esta información.

```
val poiIcon = ResourcesCompat.getDrawable(resources, R.drawable.marker_default, nu  
val poiMarker = Marker(mapView)  
poiMarker.title = "La Plata."  
poiMarker.snippet = "Ciudad de La Plata "  
poiMarker.position = Somewhere  
poiMarker.icon = poiIcon  
poiMarker.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM)  
mapView.overlays.add(poiMarker)
```

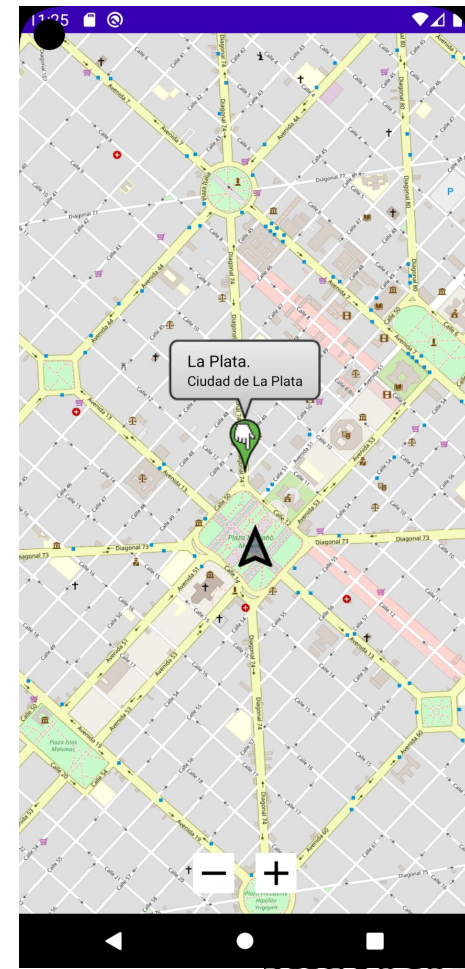
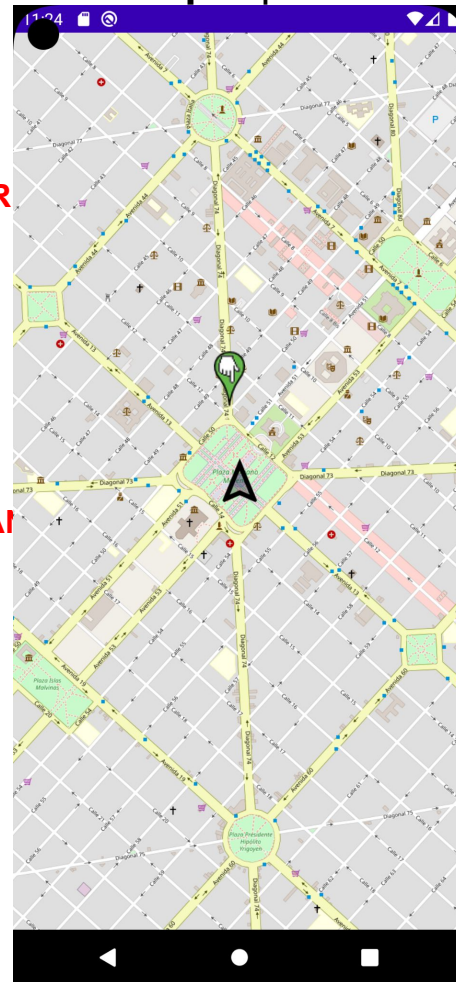


OSMDroid

Markers

Para **añadir marcas** en puntos específicos se debe **definir una lista de puntos** con una **serie de atributos** particulares y **crear una nueva capa** que contenga esta información.

```
val poiIcon = ResourcesCompat.getDrawable(resources, R.drawable.poiIcon, null)
val poiMarker = Marker(mapView)
poiMarker.title = "La Plata."
poiMarker.snippet = "Ciudad de La Plata "
poiMarker.position = Somewhere
poiMarker.icon = poiIcon
poiMarker.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM)
mapView.overlays.add(poiMarker)
```



OSMBonusPack

GeocoderNominatim y Marker Clustering

- **GeocoderNominatim** es una **clase** que es equivalente al **Geocoder** de Android. En lugar de utilizar el servidor de Google, utiliza el servidor ***nominatim*** de **OpenStreetMap** u otro especificado por el desarrollador.
- **Nominatim** es una **herramienta** que se utiliza para **buscar un punto geográfico** a partir de un nombre de **calle, pueblo, país, lugar**, etc.

OSMBonusPack

GeocoderNominatim y Marker Clustering

```

val poiProvider = NominatimPOIProvider("OsmNavigator/1.0")
val executor = Executors.newFixedThreadPool(1)
val TareaPois = FutureTask( {
    val poiProvider = NominatimPOIProvider("OSMBonusPackTutoUserAgent")
    val pois = poiProvider.getPOICloseTo(Somewhere, "cinema", 50, 0.1)
    val poiMarkers = RadiusMarkerClusterer(this@MainActivity)
    val clusterIcon: android.graphics.Bitmap? =
        BitmapFactory.decodeResource( resources, R.drawable.marker_cluster)
    poiMarkers.setIcon(clusterIcon)
    poiMarkers.textPaint.textSize = 12 * resources.displayMetrics.density
    poiMarkers.mAnchorV = Marker. ANCHOR_BOTTOM
    poiMarkers.mTextAnchorU = 0.70f
    poiMarkers.mTextAnchorV = 0.27f
    mapView.getOverlays().add(poiMarkers)
    if (pois != null) {
        for (poi in pois) {
            val poiMarker2 = Marker(mapView)
            poiMarker2.title = poi.mType
            poiMarker2.snippet = poi.mDescription
            poiMarker2.position = poi.mLocation
            poiMarker2.setAnchor(Marker. ANCHOR_CENTER, Marker. ANCHOR_BOTTOM)
            if (poi.mThumbnail != null) {
                poiMarker2.image = BitmapDrawable( resources, poi.mThumbnail)
            }
            poiMarker2.relatedObject = poi
            poiMarkers.add(poiMarker2)
        }
    }
}, "")
executor.submit(TareaPois)

```

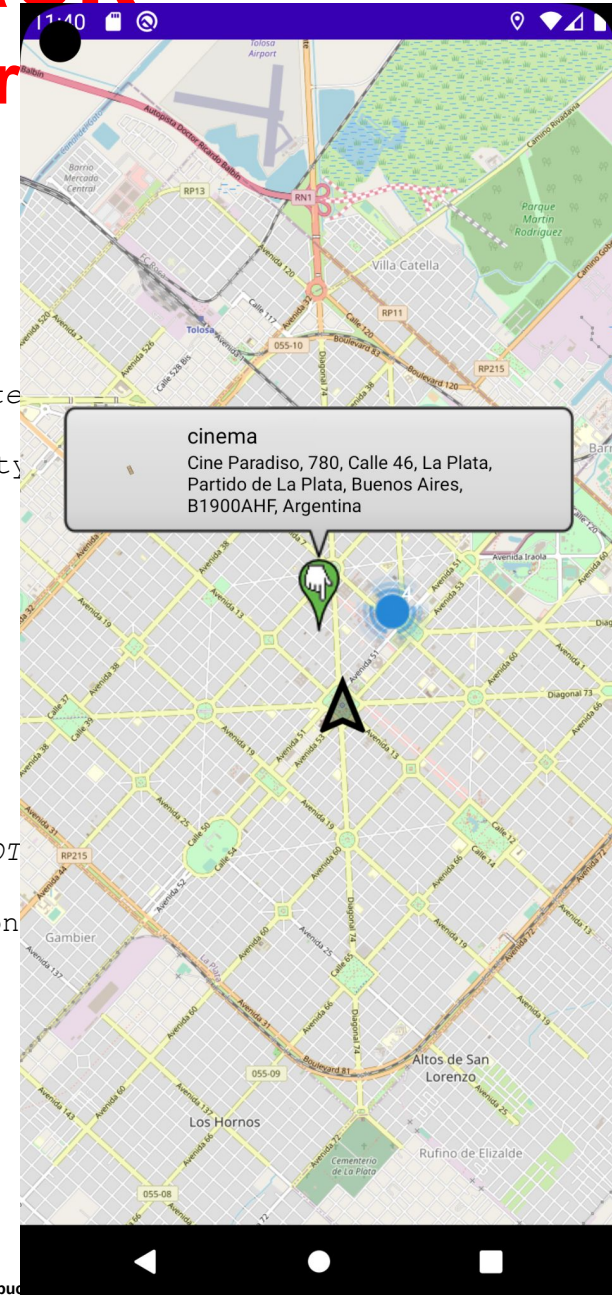

OSMBonusPack

GeocoderNominatim y Marker

```

val poiProvider = NominatimPOIProvider("OsmNavigator/1.0")
val executor = Executors.newFixedThreadPool(1)
val TareaPois = FutureTask( {
    val poiProvider = NominatimPOIProvider("OSMBonusPackTutoUserAgent")
    val pois = poiProvider.getPOICloseTo(Somewhere, "cinema", 50, 0.1)
    val poiMarkers = RadiusMarkerClusterer(this@MainActivity)
    val clusterIcon: android.graphics.Bitmap? =
        BitmapFactory.decodeResource( resources, R.drawable.marker_cluster)
    poiMarkers.setIcon(clusterIcon)
    poiMarkers.textPaint.textSize = 12 * resources.displayMetrics.density
    poiMarkers.mAnchorV = Marker.ANCHOR_BOTTOM
    poiMarkers.mTextAnchorU = 0.70f
    poiMarkers.mTextAnchorV = 0.27f
    mapView.getOverlays().add(poiMarkers)
    if (pois != null) {
        for (poi in pois) {
            val poiMarker2 = Marker(mapView)
            poiMarker2.title = poi.mType
            poiMarker2.snippet = poi.mDescription
            poiMarker2.position = poi.mLocation
            poiMarker2.setAnchor(Marker.ANCHOR_CENTER, Marker.ANCHOR_BOTTOM)
            if (poi.mThumbnail != null) {
                poiMarker2.image = BitmapDrawable( resources, poi.mThumbnail)
            }
            poiMarker2.relatedObject = poi
            poiMarkers.add(poiMarker2)
        }
    }
}, "")
executor.submit(TareaPois)

```



OSMBonusPack

OSRMRoadManager

- **OSRMRoadManager** es la **clase** encargada de **obtener la ruta desde un origen hacia un destino**, la cual es calculada en el **servidor de OSM** o de otro servidor similar para el cual se está orientando la aplicación.
- Las **rutas son devueltas** en forma de indicaciones previamente definidas como "ve a la calle 'x' o gira a la derecha".
- Las **rutas se construyen** a partir de objetos **Road**, **RoadNode** y **RoadLeg**, los cuales todos juntos definen la ruta.

OSMBonusPack

OSRMRoadManager

```
val executor: ExecutorService = Executors.newFixedThreadPool(1)

val futureTask: FutureTask<String> = FutureTask( Runnable {
    val roadManager: RoadManager = OSRMRoadManager(this@MainActivity, "")
    val waypoints = ArrayList<GeoPoint>()
    waypoints.add(Somewhere)
    val endPoint = GeoPoint(-34.91204, -57.95554)
    waypoints.add(endPoint)
    val road = roadManager.getRoad(waypoints)
    if (road.mStatus != Road.STATUS_OK) Toast.makeText(
        this@MainActivity,
        "Error!!!",
        Toast.LENGTH_SHORT
    ).show()
    val roadOverlay: Polyline = RoadManager.buildRoadOverlay(road)
    mapView.getOverlays().add(roadOverlay)
}, "")

executor.submit(futureTask)
```

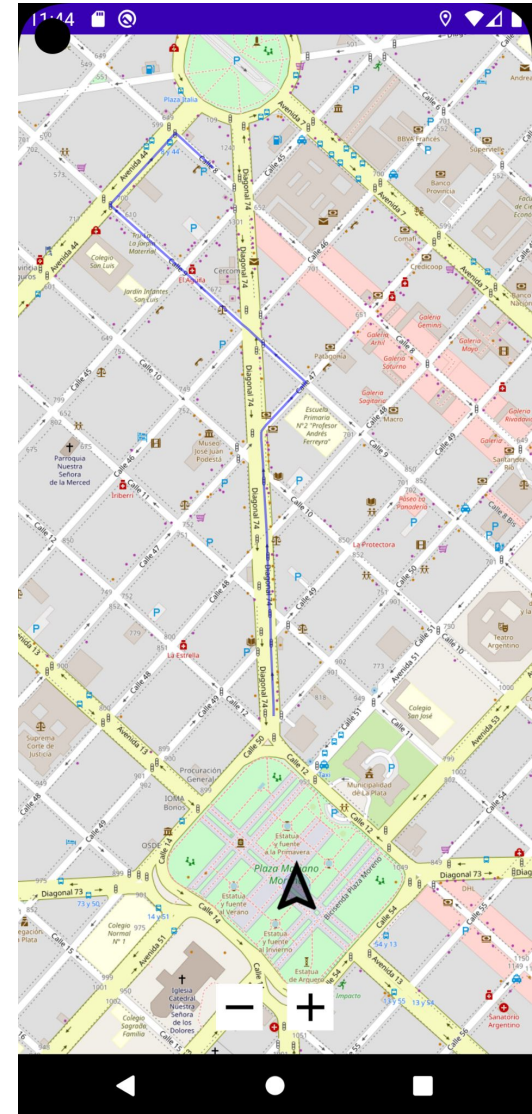
OSMBonusPack

OSRMRoadManager

```

val executor: ExecutorService = Executors.newFixedThreadPool(1)
val futureTask: FutureTask<String> = FutureTask( Runnable {
    val roadManager: RoadManager = OSRMRoadManager(this@MainActivity, "")
    val waypoints = ArrayList<GeoPoint>()
    waypoints.add(Somewhere)
    val endPoint = GeoPoint(-34.91204, -57.95554)
    waypoints.add(endPoint)
    val road = roadManager.getRoad(waypoints)
    if (road.mStatus != Road.STATUS_OK) Toast.makeText(
        this@MainActivity,
        "Error!!!",
        Toast.LENGTH_SHORT
    ).show()
    val roadOverlay: Polyline = RoadManager.buildRoadOverlay(road)
    mapView.getOverlays().add(roadOverlay)
}, "")
executor.submit(futureTask)

```



Referencias

Sitios:

<https://www.openstreetmap.org/>

<https://wiki.openstreetmap.org/wiki/Android>

<https://github.com/osmdroid/osmdroid/wiki/>

<https://www.dosmweb.com/blog/index.php?post/2016/03/28/OpenStreetMap-en-Android-Studio>

Video:

OpenStreetMap for Android: <https://www.youtube.com/watch?v=sNnMUV-Hsuw>