CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 07

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES 24/25
CFGS DAM

LIBRERÍAS MULTIMEDIA (PARTE 2)

IMPLEMENTACIÓN DE ELEMENTOS DE UBICACIÓN EN ANDROID STUDIO

Autor: Mara Vañó
m.vanoalonso@edu.gva.es
Fecha: 2024/2025

Índice

1. Objetivos de la unidad

- Incorporar mapas
- Agregar marcadores
- Editar marcadores
- Función de ubicación y mapas

2. Agregar mapas a tu app

En esta parte de la unidad vamos a prender a añadir mapas detallados que proporciona Google. Se puede identificar ubicaciones con marcadores personalizados; aumentar los datos del mapa con superposiciones de imágenes; incorporar uno o varios mapas como fragmentos, entre otras.

La **API de Google Maps** para Android te permite incluir mapas e información personalizada de mapeos en tu app.

Con la versión 2 de la API de Google Maps para Android, se puede incorporar mapas en una actividad como un fragmentos, con un fragmento XML simple. La nueva versión de Maps ofrece funciones interesantes como mapas en 3D, mapas de interiores, híbridos y de terreno; mosaicos basados en vectores para dibujos y almacenamiento en caché más eficaces; transiciones animadas; y mucho más.

2.1 Agrega un objeto de mapa.

El SDK de Maps para Android proporciona varias clases que tu app puede usar para administrar el ciclo de vida, la funcionalidad y los datos de un mapa.

Las clases admiten interacciones de los usuarios según el modelo de IU de Android, como configurar el estado inicial del mapa y responder a la entrada de gestos de los usuarios en el tiempo de ejecución.

A continuación, se incluyen la interfaz y las clases principales para controlar los mapas:

- **GoogleMap:** Es el punto de entrada para administrar las funciones y los datos del mapa subyacente. Tu app solo puede acceder a un objeto GoogleMap si este se recupera desde un objeto SupportMapFragment o MapView.
- **SupportMapFragment:** Es un fragmento que permite administrar el ciclo de vida de un objeto GoogleMap.
- **MapView:** Es una vista que permite administrar el ciclo de vida de un objeto GoogleMap.
- **OnMapReadyCallback:** Es una interfaz de devolución de llamada que controla los eventos y las interacciones de los usuarios del objeto GoogleMap.

Un objeto **GoogleMap** realiza estas operaciones automáticamente:

- Conexión al servicio de Google Maps
- Descarga de mosaicos de mapa
- Visualización de mosaicos en la pantalla del dispositivo
- Visualización de varios controles, como el desplazamiento lateral y el zoom
- Respuesta a gestos de desplazamiento lateral y zoom a través del movimiento del mapa y su acercamiento o alejamiento

Para utilizar un objeto GoogleMap en tu app, debes usar un objeto **SupportMapFragment** o **MapView** como contenedor del mapa, y luego recuperar el objeto GoogleMap del contenedor.

Dado que las clases de contenedor se derivan de un fragmento o una vista de Android, proporcionan al mapa las funciones de IU y administración del ciclo de vida de sus clases base de Android. La clase SupportMapFragment es el contenedor más común y moderno para un objeto GoogleMap.

2.2 Pasos a seguir

En esta sección, se explica cómo agregar un mapa básico mediante el uso de un fragmento como contenedor de mapa. No obstante, también puedes usar una vista. Para ver un ejemplo, consulta RawMapViewDemoActivity en GitHub.

https://github.com/googlemaps/android-samples/blob/main/ApiDemos/java/app/src/gms/java/com/example/mapdemo/RawMapViewDemoActivity.java

1. Para acceder al SDK, obtener una clave de API y agregar los frameworks necesarios, sigue los pasos que figuran en estas secciones:
   - Configuración en Google Cloud Console
   - Usa una clave de API
   - Cómo configurar un proyecto de Android Studio

2. Agrega un objeto **SupportMapFragment** a la actividad que controlará el mapa. Puedes agregar el fragmento de forma estática o dinámica.
3. Implementa la interfaz **OnMapReadyCallback**.
4. Establece el archivo de diseño como la vista de contenido.
5. Si agregaste el fragmento de forma estática, obtén un handle para este.
6. Registra la devolución de llamada.
7. Obtén un handle para el objeto GoogleMap.

Después, puedes agregar un objeto SupportMapFragment a la app de forma estática o dinámica. La manera más sencilla es hacerlo de manera estática. No obstante, si agregas el fragmento de forma dinámica, podrás realizar acciones adicionales, como quitarlo y reemplazarlo durante el tiempo de ejecución.

Cómo agregar un fragmento de forma estática:

En el archivo de diseño de la actividad que controlará el mapa, sigue estos pasos:

1. Agrega un elemento fragment.
2. Agrega la declaración de nombre xmlns:map="http://schemas.android.com/apk/res-auto". Esto habilita el uso de atributos XML personalizados de maps.
3. En el elemento fragment, configura el atributo android:name como com.google.android.gms.maps.SupportMapFragment.
4. En el elemento fragment, agrega el atributo android:id y configúralo en el ID de recurso R.id.map (@+id/map).

Por ejemplo, aquí te mostramos un archivo de diseño completo que incluye un elemento fragment:

```
<fragment xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:map="http://schemas.android.com/apk/res-auto"
android:name="com.google.android.gms.maps.SupportMapFragment"
android:id="@+id/map"
android:layout_width="match_parent"
android:layout_height="match_parent"/>
```

Recordad que hay que instalar la librería que contiene los mapas, si añades el mapa desde el layout Android te lo sugiera directamente.

En la actividad:

- Crea una instancia de SupportMapFragment.
- Confirma una transacción que agregue el fragmento a la actividad. Para obtener más información, consulta Transacciones de fragmentos.

Por ejemplo:

```
val mapFragment = SupportMapFragment.newInstance()
supportFragmentManager
.beginTransaction()
.add(R.id.my_container, mapFragment)
.commit()
```

A continuación, se debe implementar interfaz **OnMapReadyCallback.**

En el método onCreate de tu actividad, llama al método setContentView y configura el archivo de diseño como la vista de contenido.

Por ejemplo, si el nombre del archivo de diseño es main.xml, el código se verá así:
Actualiza la declaración de actividad de la siguiente manera:

```
package com.example.myapplication

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.google.android.gms.maps.GoogleMap
import com.google.android.gms.maps.OnMapReadyCallback

class MainActivity : AppCompatActivity(), OnMapReadyCallback {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

    }
```

El siguiente paso es obtener un handle para el fragmento y registrar la devolución de llamada.

- A fin de obtener un handle para el fragmento, llama al método **FragmentManager.findFragmentById** y pásale el ID de recurso del fragmento en tu archivo de diseño. Si agregaste el fragmento de forma dinámica, omite este paso porque ya recuperaste el handle.

- Llama al método getMapAsync para configurar la devolución de llamada en el fragmento.

Por ejemplo, si agregaste el fragmento de forma estática, el código se verá así:

```
val mapFragment = supportFragmentManager
    .findFragmentById(R.id.map) as SupportMapFragment
mapFragment.getMapAsync(this)
```

Utiliza el método de devolución de llamada onMapReady a fin de obtener un handle para el objeto GoogleMap. La devolución de llamada se activa cuando el mapa está listo para recibir entradas del usuario. Proporciona una instancia no nula de la clase GoogleMap que puedas usar para actualizar el mapa.

En este ejemplo, la devolución de llamada onMapReady recupera un handle para el objeto GoogleMap y, luego, se agrega un marcador al mapa:

```
override fun onMapReady(googleMap: GoogleMap) {
    googleMap.addMarker(
        MarkerOptions()
            .position(LatLng(0.0, 0.0))
            .title("Marker")
    )
}
```

Cuando compiles y ejecutes la app correctamente, se mostrará un mapa con un marcador en una isla nula (latitud y longitud de cero grados).

3. Marcadores

Los marcadores indican ubicaciones únicas en el mapa. Si deseas personalizar los marcadores, puedes cambiar el color predeterminado o reemplazar su ícono por una imagen personalizada. Las ventanas de información pueden proporcionarle contexto adicional a un marcador.

El repositorio ApiDemos en GitHub incluye un ejemplo que muestra varias características de los marcadores:

· **MapWithMarker:** Un mapa simple con un marcador (consulta el instructivo sobre cómo agregar un mapa con un marcador)

· **MarkerDemoActivity:** Uso de marcadores en un mapa, incluidos objeto de escucha y opciones.

Los marcadores identifican ubicaciones en el mapa. El marcador predeterminado utiliza un ícono estándar, que refleja el estilo de Google Maps. Es posible cambiar el color, la imagen o el punto de anclaje del ícono a través de la API. Los marcadores son objetos de tipo Marker y se agregan al mapa con el método GoogleMap.addMarker(markerOptions).

Los marcadores están diseñados para ser interactivos. Reciben eventos de clic (click) de forma predeterminada y, a menudo, se utilizan con los objetos de escucha de eventos para mostrar ventanas de información. Si la propiedad draggable de un marcador se configura en true, el usuario podrá cambiar su posición. Para ello, deberás mantenerlo presionado y arrastrarlo.

De forma predeterminada, cuando un usuario presiona un marcador, aparece la barra de herramientas del mapa en la esquina inferior derecha. Esta barra de herramientas le proporciona al usuario un acceso rápido a la app para dispositivos móviles de Google Maps. Puedes inhabilitarla si lo deseas. Para obtener más información, consulta la guía de controles.

Vídeo explicativo como agregar marcadores:

https://www.youtube.com/watch?v=l3bw8Senjmg

En el siguiente ejemplo, se muestra cómo agregar un marcador a un mapa. El marcador se crea en las coordenadas -33.852,151.211 (Sídney, Australia) y muestra la string "Marcador en Sídney" en una ventana de información cuando se hace clic en él.

```
override fun onMapReady(googleMap: GoogleMap) {
    // Add a marker in Sydney, Australia,
    // and move the map's camera to the same location.
    val sydney = LatLng(-33.852, 151.211)
    googleMap.addMarker(
        MarkerOptions()
            .position(sydney)
            .title("Marker in Sydney")
    )
    googleMap.moveCamera(CameraUpdateFactory.newLatLng(sydney))
}
```

Asocia datos a un marcador

Utiliza el método Marker.setTag() para almacenar un objeto de datos arbitrario con un marcador y el método Marker.getTag() para recuperar dicho objeto. En el ejemplo que se incluye a continuación, se muestra cómo puedes contar la cantidad de veces que se hizo clic en un marcador mediante el uso de etiquetas:

```
override fun onMapReady(map: GoogleMap) {
    // Add some markers to the map, and add a data object to each marker.
    markerPerth = map.addMarker(
        MarkerOptions()
            .position(PERTH)
            .title("Perth")
    )
    markerPerth?.tag = 0
    markerSydney = map.addMarker(
        MarkerOptions()
            .position(SYDNEY)
            .title("Sydney")
    )
    markerSydney?.tag = 0
    markerBrisbane = map.addMarker(
        MarkerOptions()
```

```
            .position(BRISBANE)
            .title("Brisbane")
        )
        markerBrisbane?.tag = 0

        // Set a listener for marker click.
        map.setOnMarkerClickListener(this)
    }

    /** Called when the user clicks a marker.  */
    override fun onMarkerClick(marker: Marker): Boolean {

        // Retrieve the data from the marker.
        val clickCount = marker.tag as? Int

        // Check if a click count was set, then display the click count.
        clickCount?.let {
            val newClickCount = it + 1
            marker.tag = newClickCount
            Toast.makeText(
                this,
                "${marker.title} has been clicked $newClickCount times.",
                Toast.LENGTH_SHORT
            ).show()
        }

        // Return false to indicate that we have not consumed the event and that we wish
        // for the default behavior to occur (which is for the camera to move such that the
        // marker is centered and for the marker's info window to open, if it has one).
        return false
    }
}
```

Video para personalizar marcador:

https://www.youtube.com/watch?v=FFm9NmSꓤw0&list=PLOU2XLYxmsIKgVkdTcsULfTIIsKz4n-I8

4. BIBLIOGRAFÍA

i. Android. Programación Multimedia y de dispositivos móviles. Garceta.

ii. Programación Multimedia y Dispositivos Móviles. Editorial Síntesis.

iii. Android App Development. For Dummies.

iv. Beginning Android Programming with Android Studio. Wrox.

v. Java Programming for Android Developers. For Dummies.

vi. https://academiaandroid.com/

vii. https://developer.android.com/

viii. https://www.redhat.com

ix. https://desarrolloweb.com

x. https://developers.google.com/maps/documentation/android-sdk/marker?hl=es-419

xi. https://developer.android.com/training/maps?hl=es-419

xii. https://developers.google.com/maps/documentation/android-sdk/marker?hl=es-419

xiii. https://developers.google.com/maps/documentation/android-sdk/map?hl=es-419

xiv. https://developers.google.com/maps/documentation/android-sdk/streetview?hl=es-419