

Software para Dispositivos Móviles
Grado en Ingeniería Informática del Software
Escuela de Ingeniería Informática – Universidad de Oviedo

Personalización de mapas en Android

Elementos, controles e interacción

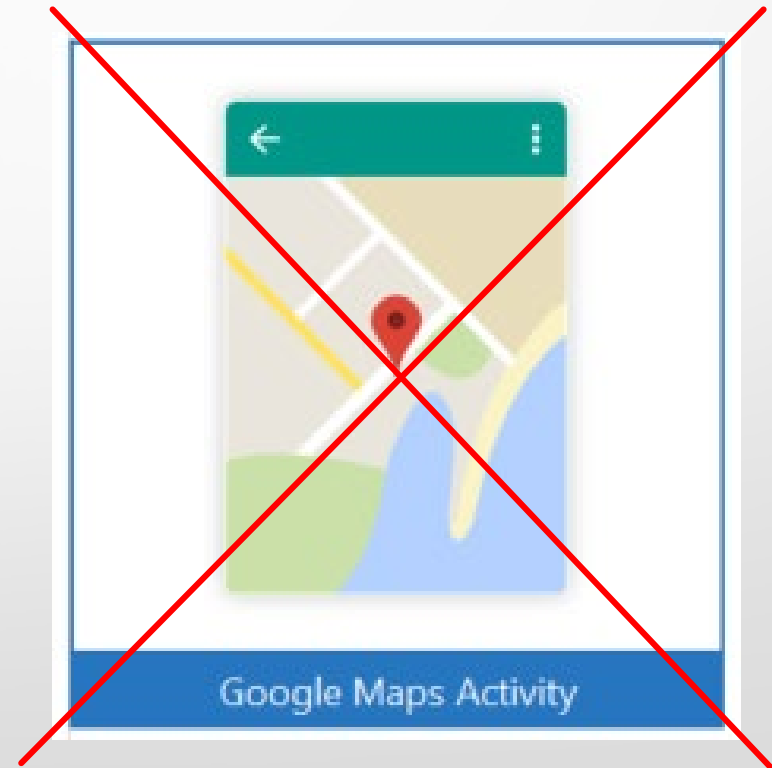
Juan Ramón Pérez Pérez

Departamento de Informática

jrpp@uniovi.es

Proyecto App Android con Mapas (Android Studio)

- Plantilla para la actividad inicial (no disponible):
 - Google Maps Activity
- También podemos crear una activity secundaria con mapas utilizando el asistente



EmptyView activity

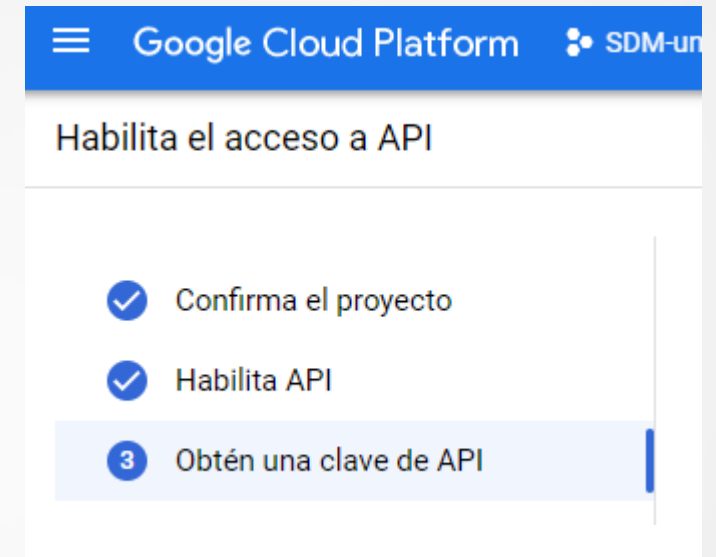
Servicio Google Maps

- Google Maps es un servicio de Google



Activar en la consola de servicios de Google ([Google Cloud Platform](#))

- Peticiones necesitan identificarse con una API Key del usuario
- Plantilla → Android manifest → url comentarios:
 - Ayuda: Cómo usar claves de API
 - [Cómo crear clave](#). URL para crear todo lo que necesitamos
- Restringir clave a Apps para Android:
 - Introducir: paquete, [firma digital sha1](#)



Incorporación de la API key en el proyecto Android

- Incluir la API key en el AndroidManifest.xml como un elemento <meta-data> dentro de <application>

```
<meta-data  
    android:name="com.google.android.geo.API_KEY"  
    android:value="${MAPS_API_KEY}" />
```

- Por seguridad dejamos el valor de la API key en local.properties

```
MAPS_API_KEY=AlzaXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX  
XXXXXXXXXX
```

Librería para Google Maps

- La librería de Google Maps forma parte de play-services
- Incluir en libs.version.toml y en el build.gradle la última versión

```
playServicesMaps = "19.0.0"
```

```
play-services-maps = { group = "com.google.android.gms", name = "play-services-maps",  
version.ref = "playServicesMaps" }
```

```
implementation(libs.play.services.maps  
)
```

Layout: MapFragment

- El XML de layout: layout/activity_maps.xml contiene un fragment específico para mapas.

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

Objeto GoogleMap (Kotlin)

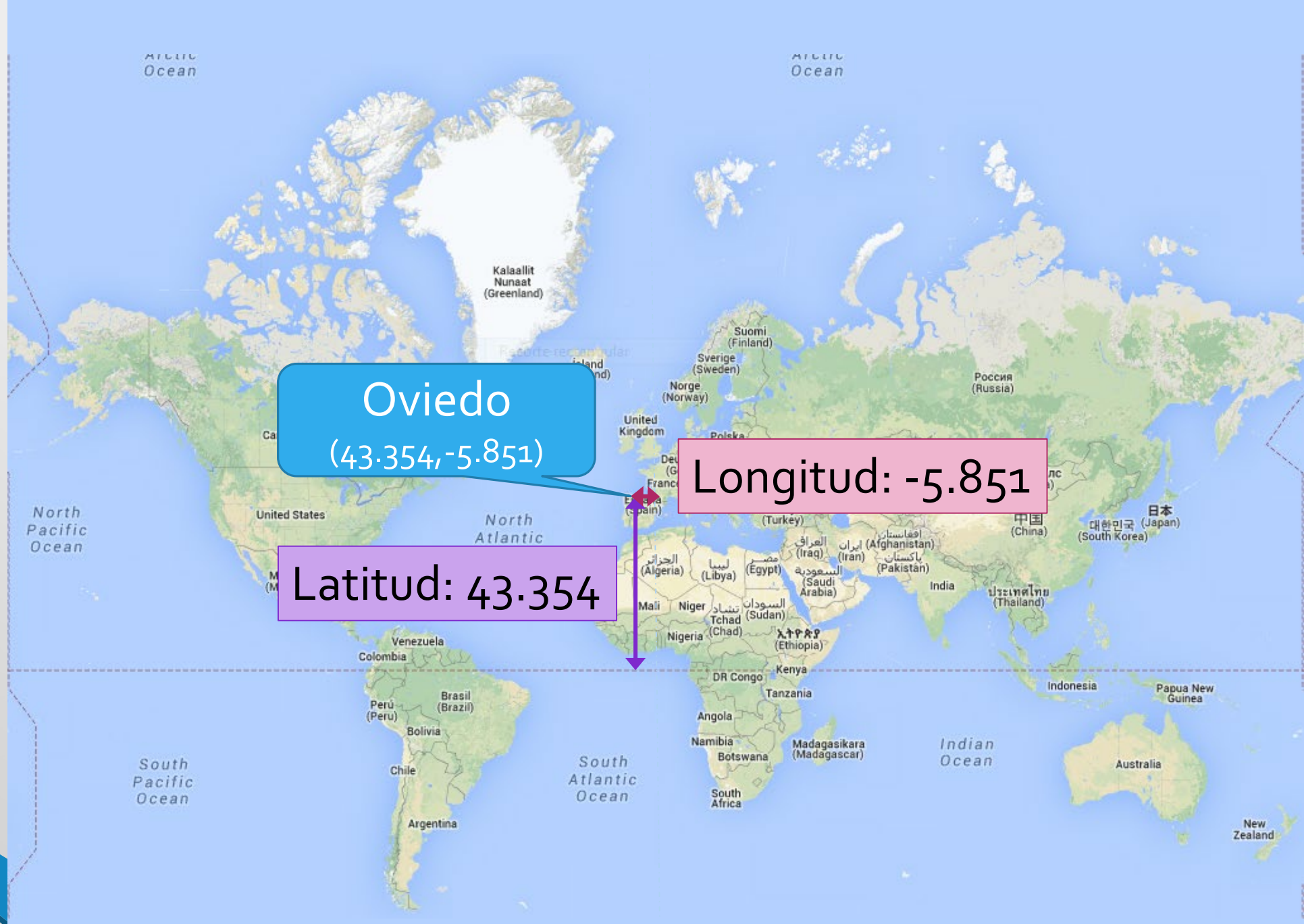
- Para obtener un objeto GoogleMap debemos de obtener el Fragment que contiene el mapa y después acceder al mapa.

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

- Para dar soporte a MapFragment necesitamos Android API 12 (3.1) o superior. Para asegurar compatibilidad hacia atrás, utilizar esta biblioteca de soporte.
- Clases de soporte: SupportMapFragment, getSupportFragmentManager()

Empezando a personalizar el mapa

- El mapa se crea mediante una llamada asíncrona
 - `mapFragment.getMapAsync(this)`
- Esperar a la función callback
 - `override fun onMapReady(googleMap: GoogleMap)`
- Para llamar a los métodos que permiten gestionar el mapa



Oviedo
(43.354,-5.851)

Longitud: -5.851

Latitud: 43.354

Cómo definimos puntos geográficos con la API de Google Maps (Kotlin)

- Utilizamos instancias de la clase LatLng
- Clase que pertenece al paquete gms.maps.model

```
val valdesSalas= LatLng(43.355115, -5.851297)
```

¿Qué elementos comprende la API de Google Maps?

Punto de
vista

Elementos
gráficos

Controles

Eventos



Implementar app: Buscar ciudades en el mapa

- La idea es buscar en un mapa, sin etiquetas, una secuencia de ciudades generada al azar.
- Para ello disponemos de una lista de ciudades con sus coordenadas y posibilidad de recuperarlas de forma aleatoria: clases *Ciudad*, *GestorCiudades*.
- Se muestra una zona de la tierra donde deben aparecer todas las ciudades de la lista, se permite al usuario señalar una ubicación y se le indica la posición real de la ciudad.

Qué vamos a hacer

Buscar ciudades en el mapa

Interfaz

- *Fragment* con el mapa
- *Button* aceptar
- *Button* siguiente
- *TextView* para el nombre de la ciudad a adivinar





Establecer el punto de vista de un mapa

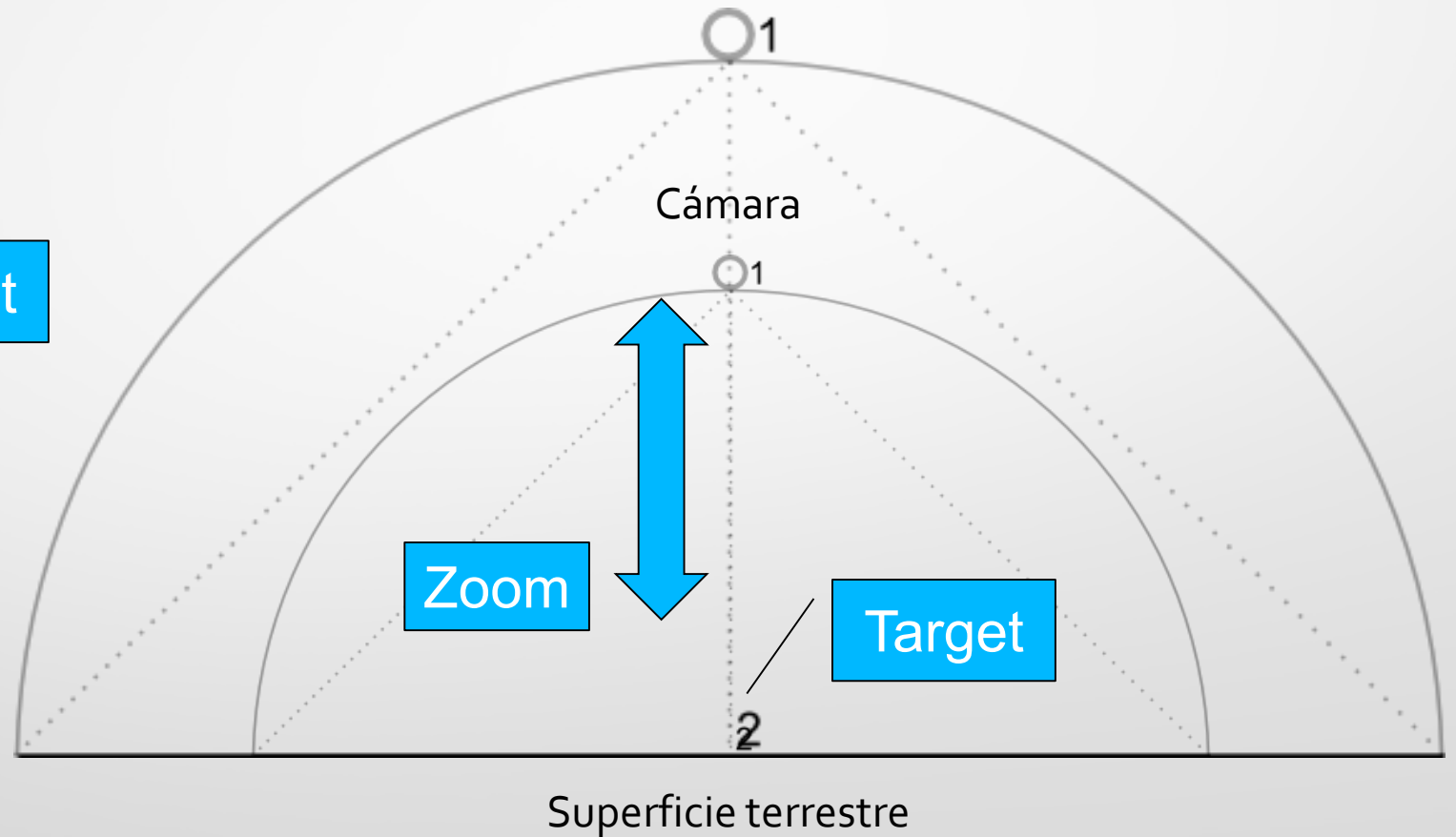
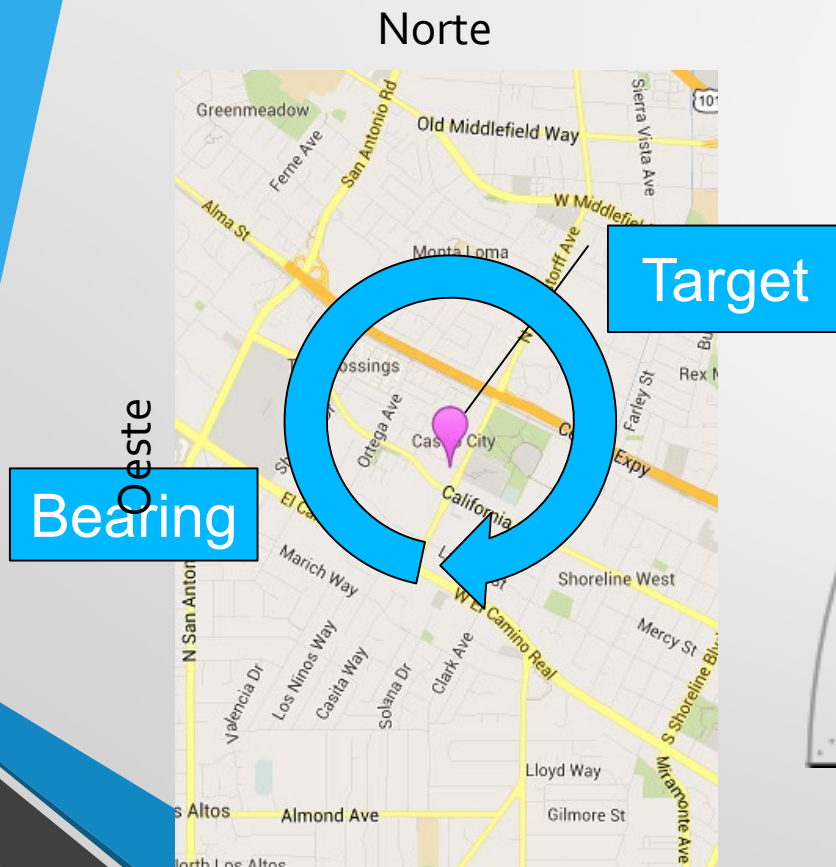
Parámetros del modelo de vista sobre un mapa

Conceptualmente disponemos de una **cámara** en la que podemos fijar con 4 parámetros:

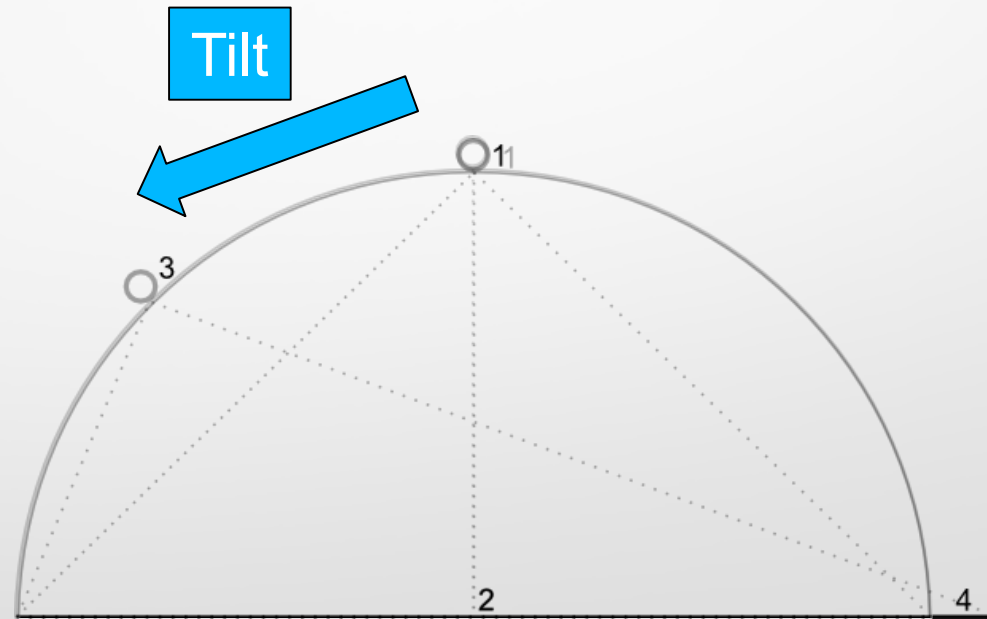
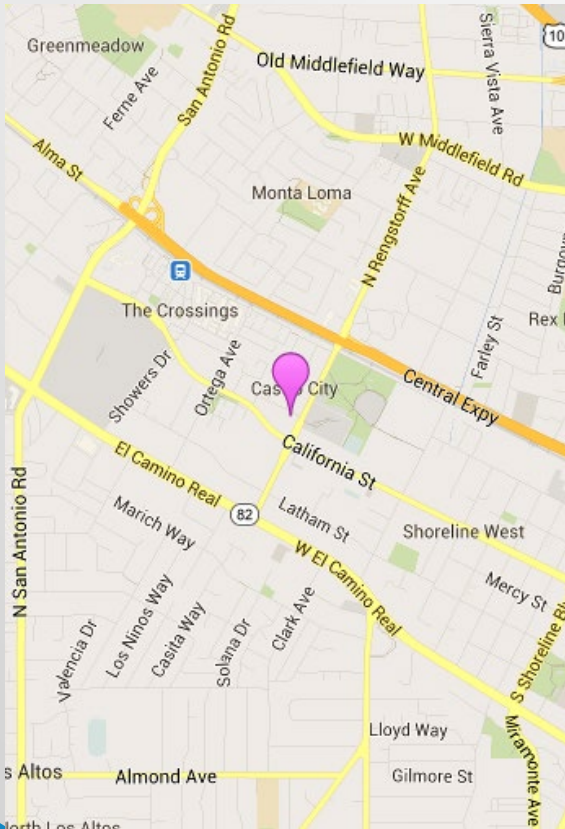
- Situar en una determinada posición geográfica (**Target**)
- Cambiar la escala, acercar y alejar (**Zoom**)
- Girar la orientación, mirar al sur (**Bearing**)
- Inclinar (**Tilt**)

<https://developers.google.com/maps/documentation/android/views>

Posicionamiento y zoom



Inclinación de la cámara



...Y con un modelo
digital de
elevaciones



CameraUpdate: ¿a dónde movemos la cámara?

- *CameraUpdate*, permite manejar todos los parámetros de la vista con la que se visualiza un mapa
- Desde el objeto GoogleMap, podemos definir como se actualiza el cambio de la cámara:
 - *moveCamera*, realiza el cambio de forma instantánea
 - *animateCamera*, realiza el cambio interpolando puntos intermedios (animación)

Factoría CameraUpdateFactory → CameraUpdate

- Nunca usamos un CameraUpdate, lo creamos a partir de [CameraUpdateFactory](#)
 - permitir fijar unos valores para los parámetros;
 - hacer variaciones sobre los actuales.

```
// Mueve la cámara instantáneamente a Oviedo con zoom 15
mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(Oviedo, 15f))
// Zoom para visualizar un rectángulo definido por las esquinas
mMap.animateCamera(CameraUpdateFactory.newLatLngBounds(peninsulaBounds, 1080, 1080, 0))
// Zoom alejándose hasta nivel 10, animación 5 segundos
mMap.animateCamera(CameraUpdateFactory.zoomTo(18f), 5000, null)
```

¿Cómo controlar todos los parámetros de la cámara a la vez? (Kotlin)

```
// Crea CameraPosition centrada en la EII y mueve la cámara hasta esa posición
val cameraPosition = CameraPosition.Builder()
    .target(valdesSalas) // Centro del mapa
    .zoom(17)           // Establece nivel de zoom
    .bearing(180)        // Orientación cámara al sur
    .tilt(30)            // Inclinação cámara 30 grados
    .build()            // Crea CameraPosition

mMap.animateCamera(
    CameraUpdateFactory.newCameraPosition(cameraPosition))
```

Configuración inicial del mapa

Permite establecer: punto de vista, tipo de mapa, controles y gestos de interacción con el mapa.

- Usando atributos XML, dentro de la definición del fragment de mapa
- Utilizando GoogleMapOptions

Se pueden invocar métodos de GoogleMap para cambiar opciones predefinidas:

- `mapa.setMapType(GoogleMap.MAP_TYPE_HYBRID);`

Google Maps también permite cambiar el estilo de los elementos del mapa

Buscar ciudades en el mapa

Punto de vista (1)

1. Situar “vista inicial” del mapa que se vea toda la península incluidas Baleares
2. Tipo de mapa satélite SIN ETIQUETAS
3. Aparece el nombre de una ciudad / población
4. (**Click Botón aceptar**) Se hace zoom para visualizar el punto de la primera ciudad a una escala mayor.
5. Siguiente recupera la “vista inicial”

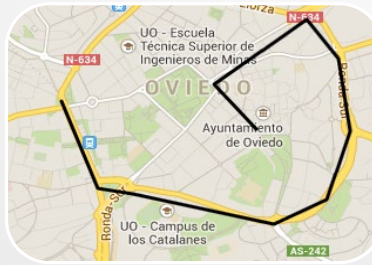


Incluir elementos gráficos en el mapa

Añadir información al mapa



Puntos, **Marker**



Lineas, **Polyline**



Zonas de mapa, **Shape**



Ventana de
información,
infoWindow

Modelo común para crear un elemento sobre el mapa

- Crear un objeto options, que recibe todas las propiedades del elemento:
 - Puntos geográficos que lo definen
 - Otras características, como el color
- Enlazarlo con el mapa mediante un método add
 - addMarker
 - addPolygon
 - ...
- Devuelve una referencia al elemento para poder manejarlo

Marcadores

Los marcadores permiten señalar puntos con unas coordenadas geográficas (latitud, longitud) dadas.

Se representan por su icono característico, que indica el punto sobre el mapa



Crear marcador (Kotlin)

```
val valdesSalas= LatLng(43.355115, -5.851297)
mMap.addMarker(MarkerOptions()
    .position(posCiudad)
    .title(binding.campoCiudad.text as String))
```

Crear marcadores personalizados (Kotlin)


Se puede personalizar el icono de los marcadores con un bitmap incluido en la aplicación o con uno creado dinámicamente

```
mapa.addMarker(MarkerOptions()  
    .icon(BitmapDescriptorFactory.fromResource(R.drawable.img))  
    .position(LatLng(43.354938, -5.852858)))
```

Crear polilíneas (Kotlin)

Una poli-línea permite establecer trayectorias en el mapa

```
val rectOptions = PolylineOptions()  
    .add(LatLng(43.355009, -5.851350))  
    .add(LatLng(43.355268, -5.851160))  
    .add(LatLng(43.356069, -5.850906))  
    .add(LatLng(43.357158, -5.851371))  
    .add(LatLng(43.357745, -5.852750))  
    .add(LatLng(43.357571, -5.853952));  
  
val polyline = mapa.addPolyline(rectOptions);
```



¿Cómo crear polígonos y círculos?

<https://developers.google.com/maps/documentation/android-api/shapes>

Gestionar elementos del mapa

- Debemos guardar la referencia al crear un elemento
 - Hacerlos visibles o invisibles: `setVisible()`
 - Eliminarlos: `remove()`
- Eliminar todos los elementos del mapa
 - `clear()`

Busca ciudades en el mapa

Inserción de elementos (2)

1. **(Click Botón aceptar)** Aparece la posición real de la ciudad con un **marcador diferente del estándar**.
2. Se trazan 3 círculos concéntricos cuyo centro es el marcador de la ciudad para dar una idea de la distancia.
3. **(Click Botón siguiente)** Se borra todos los elementos del mapa
4. Se pasa a la siguiente ciudad / población
5. (volvemos a poner otra ciudad y comenzamos ciclo, hasta que finalizamos ciudades).



Personalizar la interacción con el mapa

Controles predefinidos

Compass

MyLocation
button

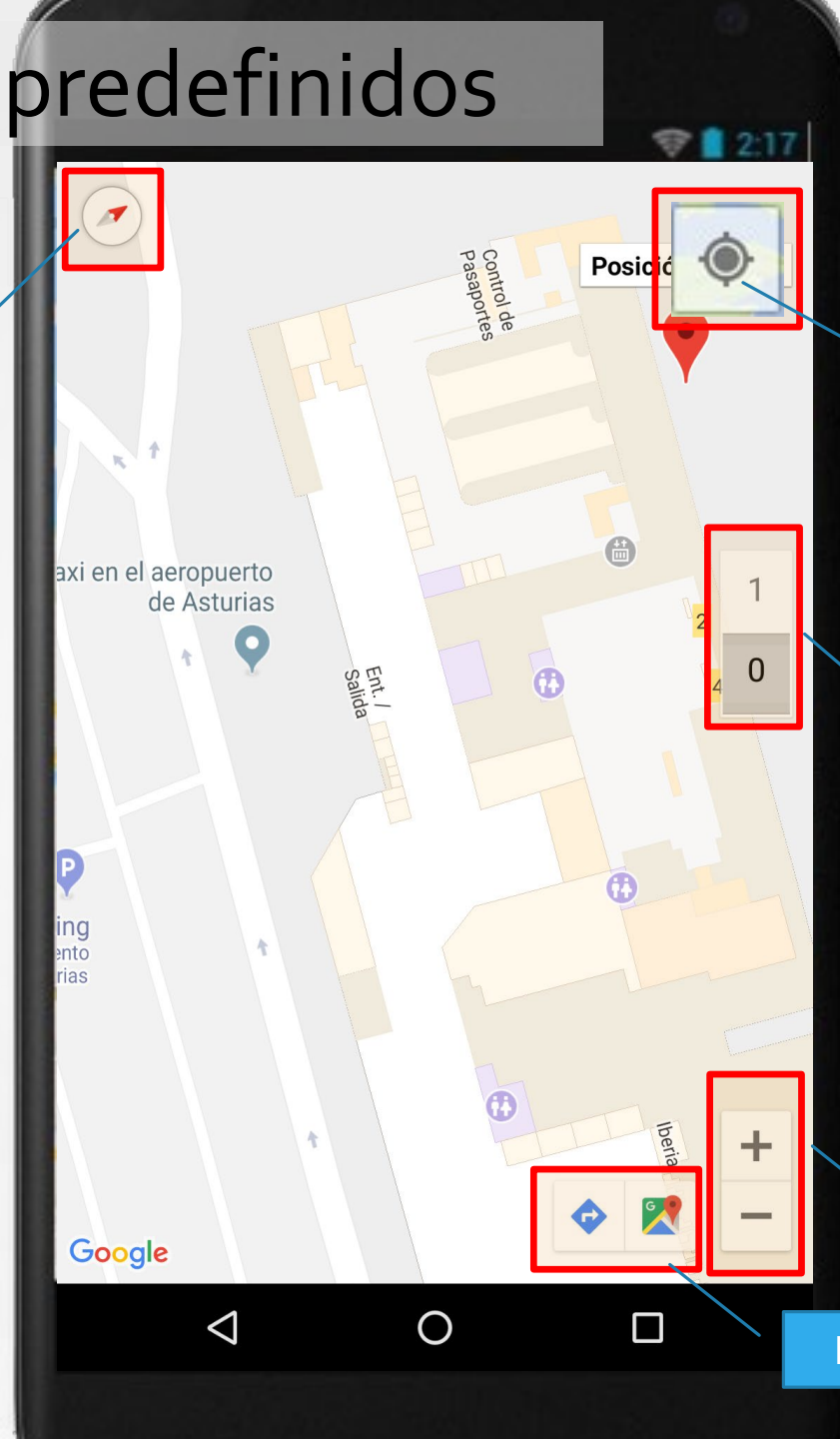
LevelPicker

ZoomControls

MapToolBar

Gestos predefinidos:

- Zoom,
- scroll,
- rotation,
- tilt



Objeto UiSettings

La API de Google Maps ofrece controles predefinidos.

La mayoría de estos controles se pueden configurar en el mapa inicial.

Se puede acceder a ellos a través de:

- `(GoogleMap)mapa.getUiSettings()` que devuelve un objeto `UiSettings`
- Desde aquí podemos activarlos u ocultarlos

Gestos sobre el mapa

El mapa creado por defecto soporta ciertos gestos para:

hacer zoom, scroll, inclinar y rotar la vista del mapa

De nuevo el objeto `UiSettings`, permite desactivar estos gestos para impedir el cambio, por parte del usuario, de alguno de estos parámetros

¿Cómo impedir que el usuario realice zoom con el gesto pero si con el control de zoom? (Kotlin)

```
// Recupera la referencia a los controles
val controles= mapa.getUiSettings()
// Muestra el control de zoom
controles.isZoomControlsEnabled= true
// Deshabilita el gesto de zoom
controles.isZoomGesturesEnabled= false
```

Qué eventos podemos capturar sobre componentes Google Maps

Google Maps dispone de varios eventos específicos sobre el mapa global o distintos componentes:

Elemento	Click	LongClick	Otros
Mapa (Map)	X	X	loaded
Marcador (Marker)	X		drag
InfoWindow	X	X	close
Polyline (clickable=true)	X		
Polygon (clickable=true)	X		
Circle (clickable=true)	X		
GroundOverlay	X		
MyLocationButton	X		
MyLocation	X		

Eventos de cambio de estado de Google Maps

- OnCameraIdleListener
- OnCameraMoveStartedListener
- OnCameraMoveListener
- OnCameraMoveCanceledListener
- OnIndoorStateChangeListener

Captura de eventos

- El desarrollador puede capturar estos eventos y asociar acciones
 - Usar los métodos del objeto GoogleMaps: `setOn<elemento><evento>Listener`
 - Ejemplo: `setOnMapLongClickListener`
 - Implementar la interfaz: `on<elemento><evento>Listener`

Caso concreto: Captura de un evento click sobre el mapa (Kotlin)

```
// Captura evento LongClick sobre el mapa
// Registra una instancia que implementa OnMapLongClickListener
mapa.setOnMapLongClickListener {
    // Crea un marcador en el punto y lo añade al mapa
    val marcadorOpciones= MarkerOptions()
        .position(punto)
        .title("Marcador creado por el usuario")
    mapa.addMarker(marcadorOpciones)
}
```

Buscar ciudades en el mapa

Eventos y controles (3)

1. Se inhabilita la posibilidad de zoom por parte del usuario: ni controles ni gestos de zoom.
2. El usuario indica donde cree que se encuentra la ciudad mediante una pulsación larga.
3. Aparece un marcador para la posición.
4. (**Botón aceptar**) Se traza una línea recta entre la posición real de la ciudad y la marcada por el usuario.
5. **Ampliación:** Permitir que el usuario rectifique la posición marcada, mientras no pulse aceptar

Medir distancias

- Clase Location (Android core)
- Sirve para dar soporte a Geolocalización
- Nos proporciona la funcionalidad para calcular distancia esférica:

```
distanceBetween(double startLatitude, double startLongitude, double  
endLatitude, double endLongitude, float[] results)
```

```
distanceTo(Location dest)
```

Busca ciudades en el mapa

Calcular distancia (4)

1. Calcular la distancia entre posición real y marcador de usuario
2. Mostrar la distancia en un InfoWindow sobre el marcador del usuario
3. Calcular una puntuación evaluando la distancia
 1. Si cae en el círculo central 10 ptos, círculo intermedio 5 ptos, círculo externo 2 ptos, fuera de los círculos 0 ptos.
4. **Ampliación:** Guardar respuestas
5. **Ampliación:** Inhabilitar botones mientras no se cumplan las condiciones para usarlos:
 - Aceptar → tiene que estar marcada una posición
 - Siguiente → tiene que haberse mostrado la posición real

Biblioteca de utilidades Maps SDK

- Existen una serie de utilidades para mapas que están en otra librería aparte de la principal:

```
implementation 'com.google.maps.android:android-maps-utils:2.1.0'
```

- Permite importar datos KML o GeoJSON como una capa del mapa
- Permite agrupar marcadores en función del zoom
- Funciones alternativas para medir distancias y áreas
- Más funcionalidades....
- <https://developers.google.com/maps/documentation/android-sdk/utility>



Práctica Completa: Buscar ciudades en el mapa

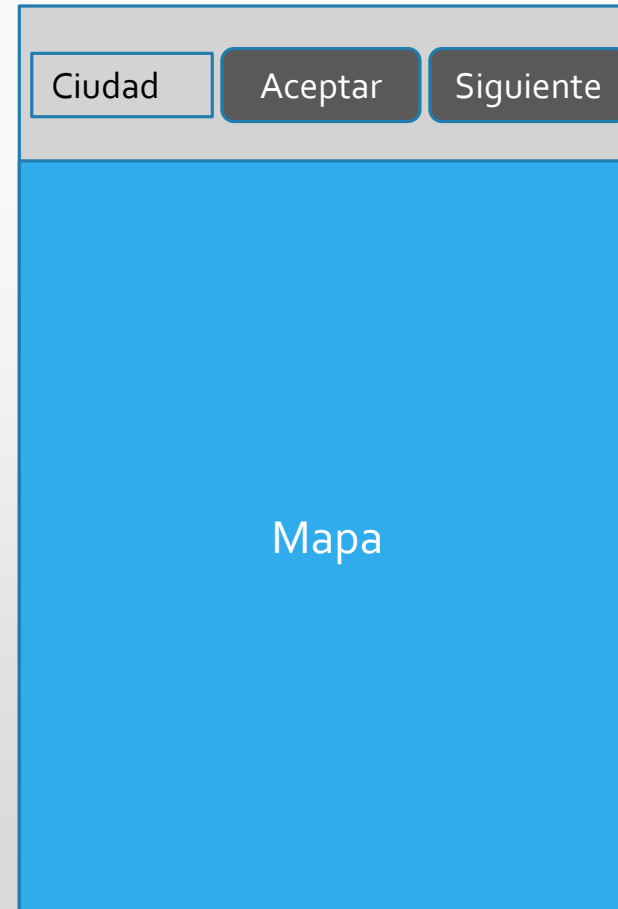
Implementar app: Buscar ciudades en el mapa

- La idea es buscar en un mapa, sin etiquetas, una secuencia de ciudades generada al azar.
- Para ello disponemos de una lista de ciudades con sus coordenadas y posibilidad de recuperarlas de forma aleatoria: clases *Ciudad*, *GestorCiudades*.
- Se muestra una zona de la tierra donde deben aparecer todas las ciudades de la lista, se permite al usuario señalar una ubicación y se le indica la posición real de la ciudad.

Buscar ciudades en la mapa

Interfaz

- *Fragment* con el mapa
- *Button* aceptar
- *Button* siguiente
- *TextView* para el nombre de la ciudad a adivinar



Buscar ciudades en el mapa

Punto de vista (1)

1. Situar vista inicial del mapa que se vea toda la península incluidas Baleares
2. Tipo de mapa satélite SIN ETIQUETAS
3. Aparece el nombre de una ciudad / población
4. (**Click Botón aceptar**) Se hace zoom para visualizar el punto de la primera ciudad a una escala mayor.

Busca ciudades en el mapa

Inserción de elementos (2)

1. **(Click Botón aceptar)** Aparece la posición real de la ciudad con un **marcador diferente del estándar**.
2. Se trazan 3 círculos concéntricos cuyo centro es el marcador de la ciudad para dar una idea de la distancia.
3. **(Click Botón siguiente)** Se borra todos los elementos del mapa
4. Se pasa a la siguiente ciudad / población
5. (volvemos a poner otra ciudad y comenzamos ciclo, hasta que finalizamos ciudades).

Buscar ciudades en el mapa

Eventos y controles (3)

1. Se inhabilita la posibilidad de zoom por parte del usuario: ni controles ni gestos de zoom.
2. El usuario indica donde cree que se encuentra la ciudad mediante una pulsación larga.
3. Aparece un marcador para la posición.
4. (**Botón aceptar**) Se traza una línea recta entre la posición real de la ciudad y la marcada por el usuario.
5. **Ampliación:** Permitir que el usuario rectifique la posición marcada, mientras no pulse aceptar

Busca ciudades en el mapa (4)

1. Calcular y mostrar la distancia en un InfoWindow sobre el marcador del usuario
2. Calcular una puntuación evaluando la distancia
 1. Si cae en el círculo central 10 ptos, círculo intermedio 5 ptos, círculo externo 2 ptos, fuera de los círculos 0 ptos.
3. **Ampliación:** Guardar respuestas
4. **Ampliación:** Inhabilitar botones mientras no se cumplan las condiciones para usarlos:
 - Aceptar → tiene que estar marcada una posición
 - Siguiente → tiene que haberse mostrado la posición real