

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

# Desarrollo de mapas personalizados en Android

*Elementos, controles e interacción*

Juan Ramón Pérez Pérez

Departamento de Informática

[jrpp@uniovi.es](mailto:jrpp@uniovi.es)

# Objeto GoogleMap


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

```
MapFragment fragMapa=  
    (MapFragment) getFragmentManager ()  
    .findFragmentById (R.id.map) ;
```

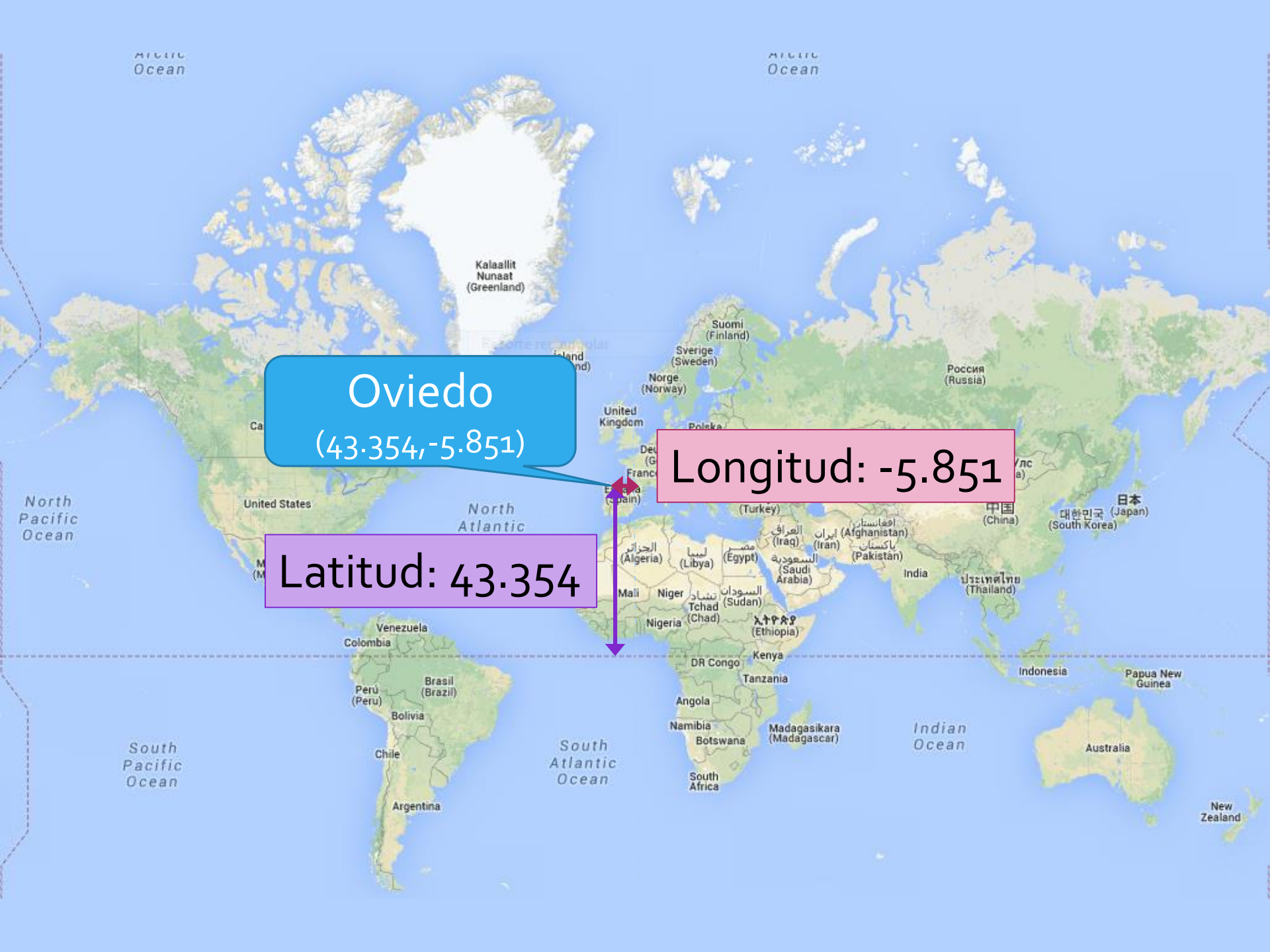
```
mapFragment.getMapAsync (this) ;
```

- Para dar soporte a MapFragment necesitamos Android API 12 (3.1) o superior, si queremos asegurar compatibilidad hacia atrás, utilizar la biblioteca de soporte.

Clases de soporte: SupportMapFragment, getSupportFragmentManager()



Elemento básico  
¿Cómo localizar un punto en el  
mapa?



Oviedo  
(43.354,-5.851)

Longitud: -5.851

Latitud: 43.354

# Cómo definimos puntos geográficos con la API de Google Maps

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

```
LatLng ValdesSalas= new LatLng(43.355115,  
-5.851297) ;
```

# ¿Qué elementos comprende la API de Google Maps?

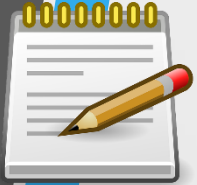
Punto de  
vista

Elementos  
gráficos

Controles

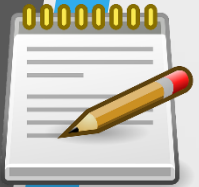
Eventos





## Implementar app: Buscar ciudades en el mapa

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

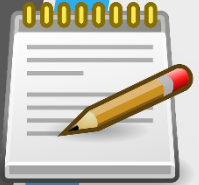


# Buscar ciudades en la mapa Layout

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

A diagram of a mobile app layout. At the top, there is a light gray header bar containing three elements: a text input field with the placeholder text "Ciudad", and two dark gray buttons with white text labeled "Aceptar" and "Siguiente". Below the header is a large blue rectangular area representing the map, with the word "Mapa" centered in white text.






# 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**) Aparece la posición real de la ciudad marcador básico.
5. Se hace zoom para visualizar el punto de la primera ciudad a una escala mayor.



Establecer el punto de vista de  
un mapa

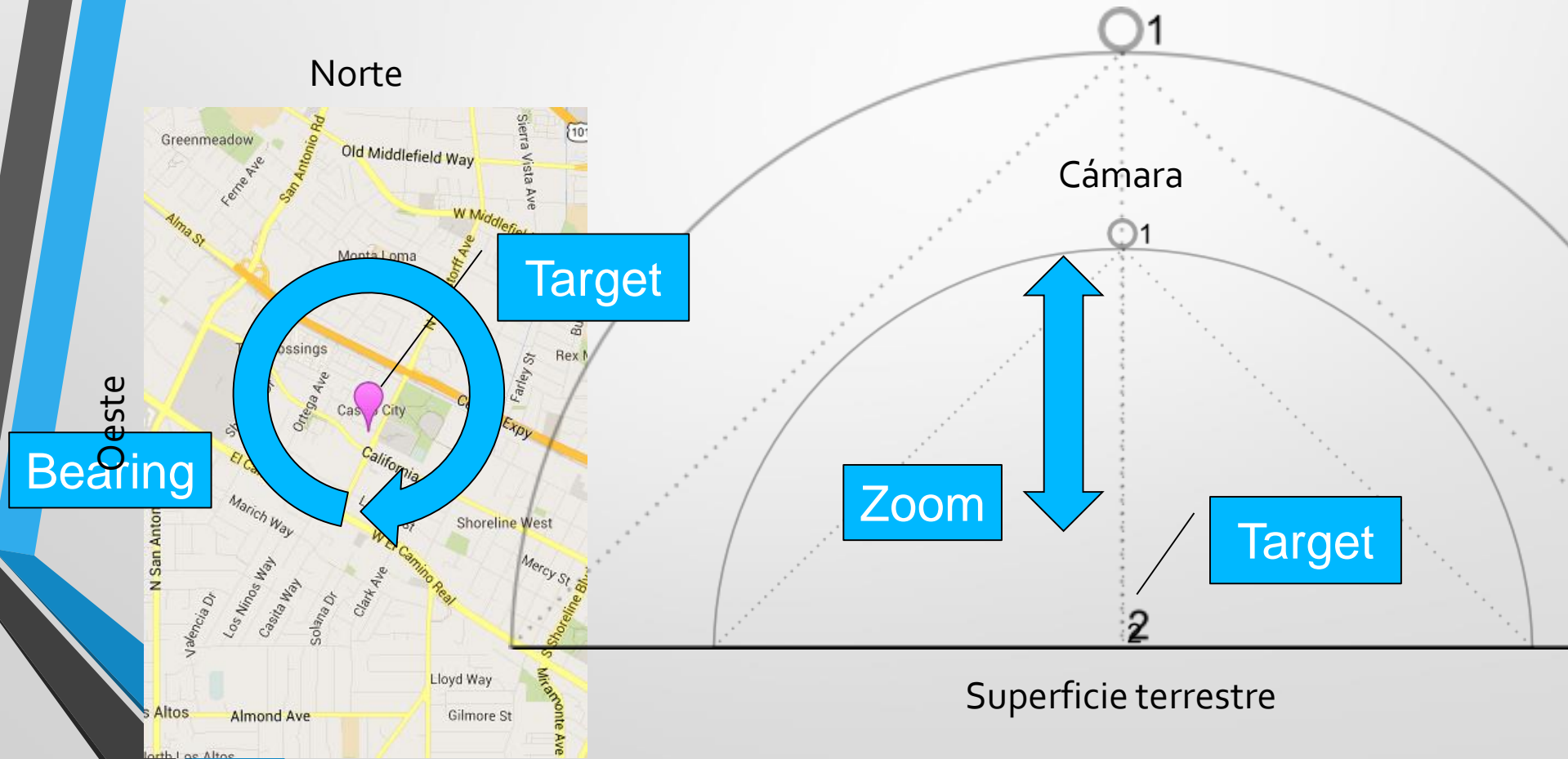
# Parámetros del modelo de vista sobre un mapa

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

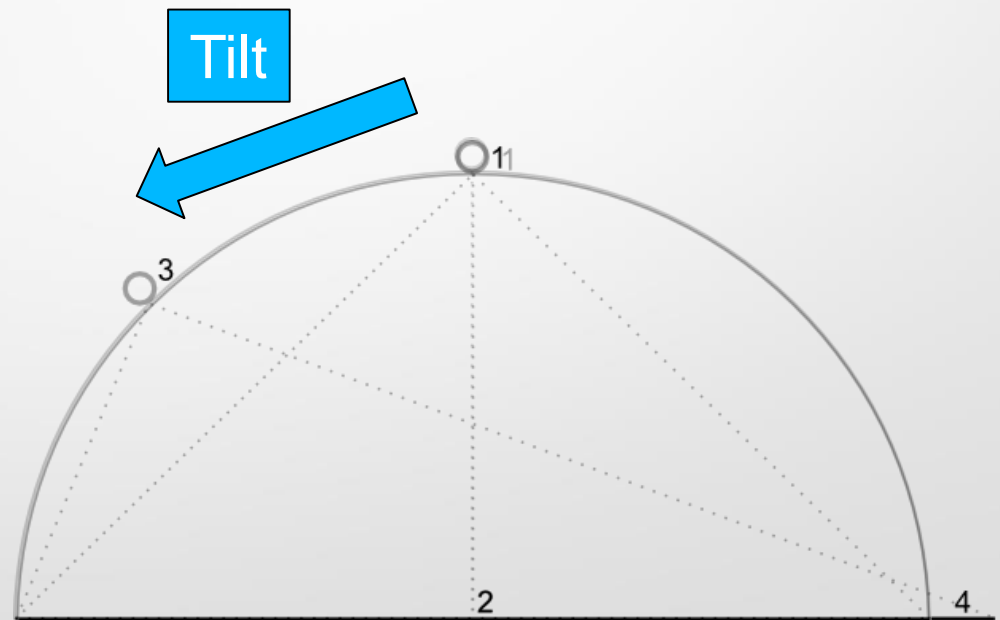
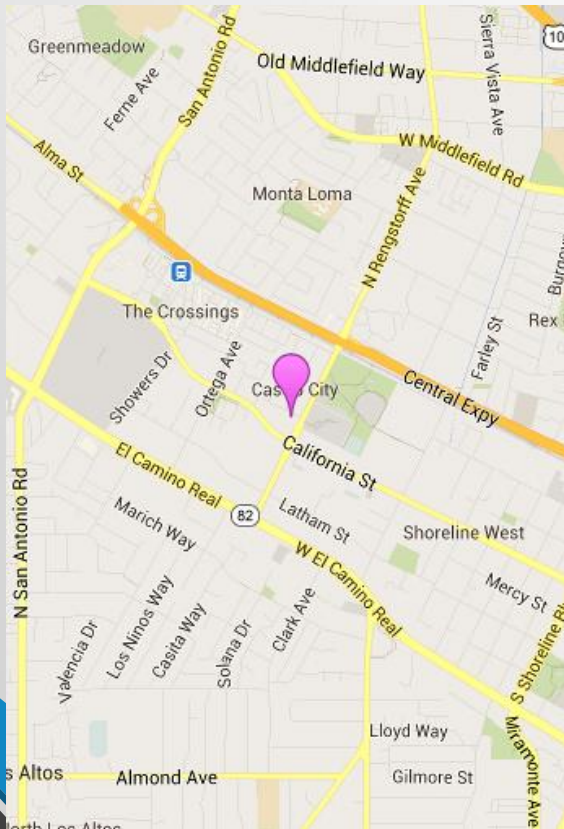
- Situar en una determinada posición geográfica (**Target**)
- Cambiar la escala, acercar y alejar (**Zoom**)
- Girar la orientación, mirar al norte (**Bearing**)
- Inclinar (**Tilt**)

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

# Posicionamiento y zoom



# Inclinación de la cámara







# 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 instantanea
  - *animateCamera*, realiza el cambio interpolando puntos intermedios (animación)

# Factoría CameraUpdateFactory → CameraUpdate

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

```
// Mueve la cámara instantáneamente al Valdés Salas con zoom 15
mapa.moveCamera(CameraUpdateFactory.newLatLngZoom(ValdesSalas,
15));
// Zoom acercándose, animando la cámara.
mapa.animateCamera(CameraUpdateFactory.zoomIn());
// Zoom alejándose hasta nivel 10, animación 5 segundos
mapa.animateCamera(CameraUpdateFactory.zoomTo(18), 5000, null);
```



# ¿Cómo controlar todos los parámetros de la cámara a la vez?

// Construye una **CameraPosition** centrada en la Escuela de Ingeniería Informática y anima la cámara hasta esa posición

```
CameraPosition cameraPosition = new
```

```
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
```

```
mapa.animateCamera (
```

```
    CameraUpdateFactory.newCameraPosition(cameraPosition));
```

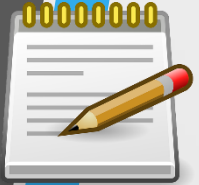
# Configuración inicial del mapa

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:

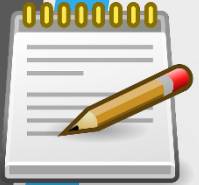
- `mapa.setMapType (GoogleMap.MAP_TYPE_HYBRID) ;`



# 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**) Aparece la posición real de la ciudad marcador básico.
5. 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. Su radio 50, 100 y 150Km
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).



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...* de la clase *GoogleMap*
  - *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

```
// Añadir marcador a un mapa
MarkerOptions marcadorOpciones= new MarkerOptions()
    .position(ValdesSalas)
    .title("Escuela de Ingeniería Informática");
Marker marcador= mapa.addMarker(marcadorOpciones);
```

# Crear marcadores personalizados

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

```
mapa.addMarker(new MarkerOptions()  
    .icon(BitmapDescriptorFactory.fromResource(R.drawable.icono_futbol  
))  
    .anchor(0.5f, 1f)  
    .position(new LatLng(43.354938, -5.852858)));
```

Más info sobre personalización de marcadores:


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

# Crear polilíneas

Una poli-línea permite establecer trayectorias en el mapa

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

```
Polyline polyline = mapa.addPolyline(rectOptions);
```

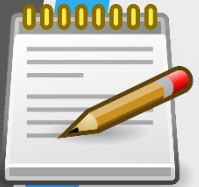


# ¿Cómo crear polígonos y círculos?

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

# Gestionar elementos del mapa

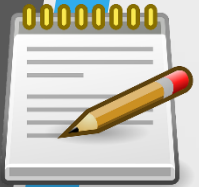
- Debemos guardar la referencia al crear un elemento
- Hacerlos visibles o invisibles: `setVisible()`
- Eliminarlos: `remove()`



# 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: no controles ni gestos de zoom.
2. El usuario indica donde cree que se encuentra la ciudad mediante una pulsación larga.
3. Aparece un marcador convencional para la posición.
4. (**Botón aceptar**) Se traza una línea recta entre la posición real de la ciudad y la marcada por el usuario.



Personalizar la interacción con  
el mapa: controles y eventos



# Controles predefinidos

Compass



My Location button



Level picker

Gestos predefinidos:

- Zoom,
- scroll,
- rotation,
- tilt

Zoom Control



# 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?

```
// Recupera la referencia a los controles
UiSettings controles= mapa.getUiSettings();
// Muestra el control de zoom
controles.setZoomControlsEnabled(true);
// Deshabilita el gesto de zoom
controles.setZoomGesturesEnabled(false);
```

Más info sobre controles y gestos:

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

# Qué eventos podemos capturar sobre un mapa

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

Más info sobre eventos:

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

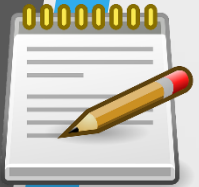
| Elemento               | Click | LongClick | Drag | Loaded |
|------------------------|-------|-----------|------|--------|
| Mapa (Map)             | X     | X         |      | X      |
| Marcador (Marker)      | X     |           | X    |        |
| InfoWindow             | X     |           |      |        |
| Cambio de cámara       |       |           |      |        |
| Mapas de interiores    |       |           |      |        |
| Formas y superposición | X     |           |      |        |

# 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

```
// Captura evento LongClick sobre el mapa
// Registra una instancia que implementa OnMapLongClickListener
mapa.setOnMapLongClickListener(new OnMapLongClickListener()
{
    // El método que recibe el callback,
    // parámetro LatLng con el punto donde se pulsó
    public void onMapLongClick(LatLng punto)
    {
        // Crea un marcador en el punto y lo añade al mapa
        MarkerOptions marcadorOpciones= new MarkerOptions()
            .position(punto)
            .title("Marcador creado por el usuario");
        mapa.addMarker(marcadorOpciones);
    }
});
```



# Buscar ciudades en el mapa

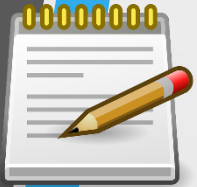
## Eventos y controles (3)

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



# Medir distancias

- Clase *Location* del paquete *android.location*
- La emplearemos en el próximo tema para guardar localizaciones
- Tiene dos métodos para medir distancias en línea recta:
  - [distanceBetween](#)
  - [distanceTo](#)



# Busca ciudades en el mapa

## Ampliaciones (4)

- Permitir que el usuario rectifique la posición marcada, mientras no pulse aceptar
- Mostrar la distancia en un InfoWindow sobre el marcador del usuario
- Calcular una puntuación evaluando la distancia
- 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