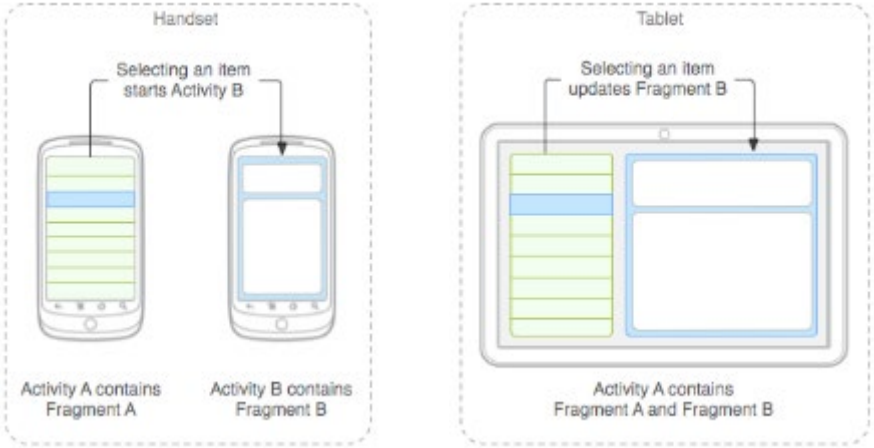
SESIÓN 7

- Fragmentos
- Mapas

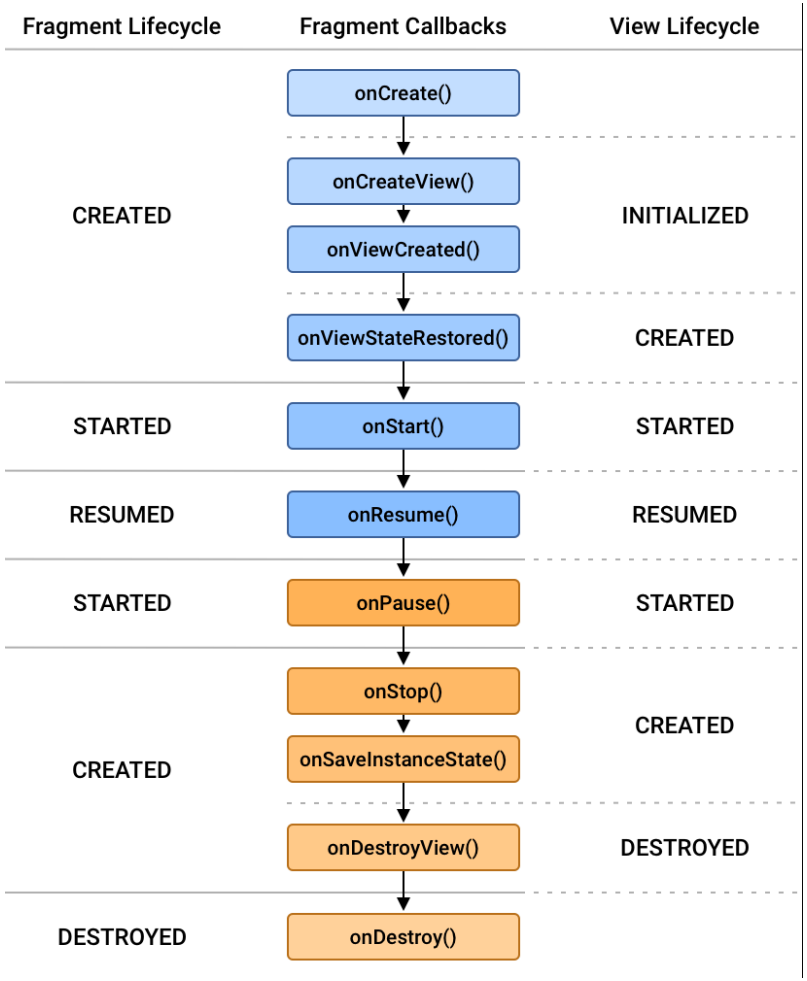
Fragmentos

- Módulo, con interfaz de usuario y comportamiento, que puede ser embebido en una actividad
- Si bien solo pueden ser usados como parte de la actividad, puede considerarse como una "subactividad" funcional, con su propio ciclo de vida
- Permiten estructurar una actividad como un conjunto de *fragmentos* aportando modularidad y reusabilidad
- Se pueden reutilizar en diferentes actividades
- Útil para hacer aplicaciones que se ejecutan en dispositivos de diferentes tamaños (teléfono, tableta, Android TV, etc)



https://developer.android.com/guide/components/fragments

Fragmentos. Ciclo de vida



https://developer.android.com/guide/fragments/lifecycle

Hilo de ejecución

- La actividad y sus fragmentos se ejecutan en el hilo de la interfaz de usuario (hilo de la GUI).
- Por tanto:
  - La ejecución del código de la actividad y sus fragmentos es lineal y secuencial
  - El fragmento puede accede a los elementos de diseño

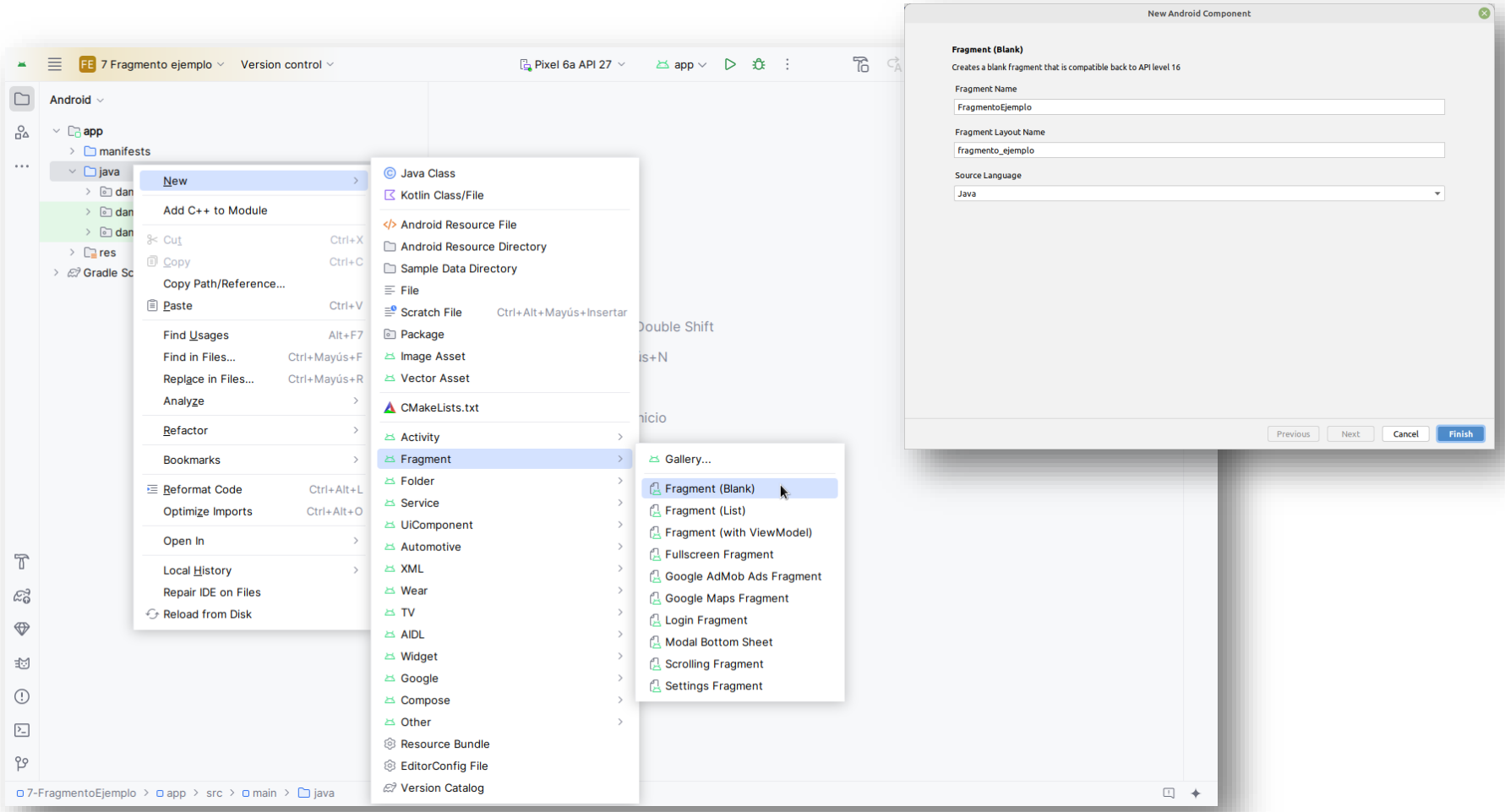
Componentes de un fragmento

- Un fragmento suele tener 2 componentes: un fichero XML de diseño (*layout*) y una clase Java para darle el comportamiento
- El fichero de diseño tiene el mismo formato que el de una actividad
- La clase Java debe ser una subclase de la clase `Fragment` (hay más) que sobrescriba, al menos, el método `onCreateView()`, responsable de cargar el diseño del fragmento. Puede sobrescribir otros métodos como `onCreate()`, `onPause()`,… `onViewCreated()`, `onDestroyView()`, …

```
class FragmentoEjemplo extends Fragment {
    @Override
    public View onCreateView (ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate (R.layout.fragmento, container, false);
    }
}
```

- La actividad que usa el fragmento debe ser una subclase de `FragmentActivity` (`AppCompatActivity` lo es)

Uso del asistente para crear fragmentos



Fichero XML de diseño del fragmento

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
  android:background="@color/design_default_color_secondary"
   tools:context=".FragmentoEjemplo">
   <!-- TODO: Update blank fragment layout -->
   <TextView
     android:layout_width=" match_parent "
     android:layout_height="match_parent"
     android:text="@string/hello_blank_fragment" />
</FrameLayout>
```

Fichero Java del fragmento

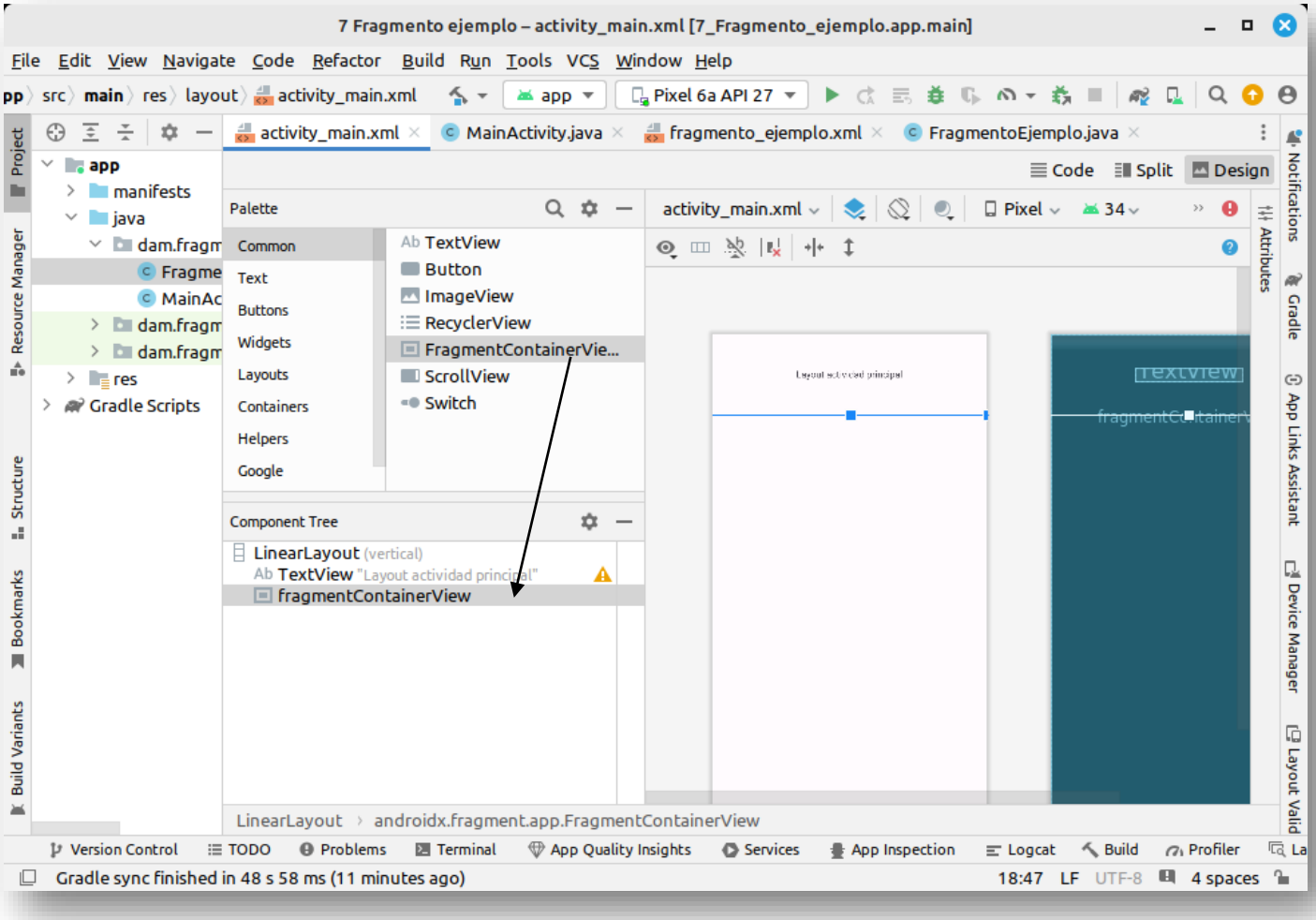
```
public class FragmentoEjemplo extends Fragment {
  // TODO: Rename parameter arguments, choose names that match
  // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
  private static final String ARG_PARAM1 = "param1";
  private static final String ARG_PARAM2 = "param2";
  // TODO: Rename and change types of parameters
  private String mParam1;
  private String mParam2;
  public FragmentEjemplo() {
    // Required empty public constructor
  }
  /**
   * Use this factory method to create a new instance of
   * this fragment using the provided parameters.
   *
   * @param param1 Parameter 1.
   * @param param2 Parameter 2.
   * @return A new instance of fragment FragmentoEjemplo.
   */
  // TODO: Rename and change types and number of parameters
  public static FragmentoEjemplo newInstance(String param1, String param2) {
    FragmentoEjemplo fragment = new FragmentoEjemplo ();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, param1);
    args.putString(ARG_PARAM2, param2);
    fragment.setArguments(args);
    return fragment;
  }
  @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      if (getArguments() != null) {
        mParam1 = getArguments().getString(ARG_PARAM1);
        mParam2 = getArguments().getString(ARG_PARAM2);
      }
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
      // Inflate the layout for this fragment
      return inflater.inflate(R.layout.fragment_ejemplo, container, false);
    }
}
```

Incorporar el fragmento a la actividad

Un fragmento se puede incorporar a la actividad:

- **Estáticamente**: añadiendo el fragmento en el fichero XML de diseño de la actividad
- **Dinámicamente**: por código cuando se necesite

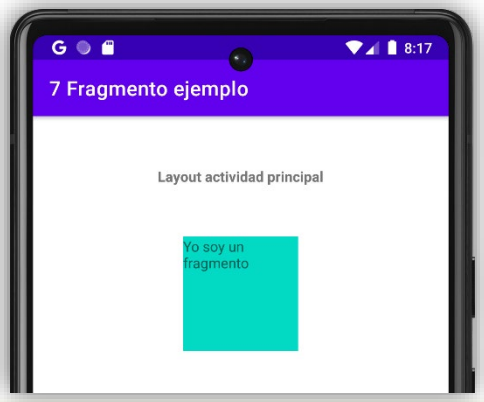
Añadir un fragmento estáticamente (1)



Añadir un fragmento estáticamente (2)

Un fragmento se puede añadir al diseño de la actividad indicándolo en el fichero XML de diseño, en un elemento de tipo <FragmentContainerView>:

- El atributo android:name debe referenciar la clase del fragmento
- El atributo tools:layout es opcional y permite ver el diseño del fragmento en la vista diseño de Android Studio




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context=".MainActivity">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
   android:layout_gravity="center"
    android:layout_margin="50dp"
    android:text="Layout actividad principal" />
  <androidx.fragment.app.FragmentContainerView
    android:id="@+id/contenedor_fragmento"
    android:name="dam.fragmentoejemplo.FragmentoEjemplo"
    android:layout_width="114dp"
    android:layout_height="114dp"
    android:layout_gravity="center"
    tools:layout="@layout/fragmento_ejemplo" />
</LinearLayout>
```

Se puede cambiar por otro contenedor, por ejemplo: Fragment, LinearLayout, …

Añadir fragmento dinámicamente (1)

- Un fragmento añadido estáticamente (en el fichero XML de diseño de la actividad) se instancia y muestra al iniciarse la actividad (aunque se puede ocultar)
- Un fragmento añadido dinámicamente (por código) puede instanciarse y mostrarse en la actividad cuando se necesite en tiempo de ejecución, y también puede eliminarse o intercambiarse por otro fragmento
- Un fragmento añadido por código también tiene su fichero XML de diseño

Añadir fragmento dinámicamente (2)

Ejemplo del fichero XML de la actividad que va a añadir el fragmento por código en el elemento contenedor <FragmentContainerView>

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context=".MainActivity">
  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
   android:layout_gravity="center"
    android:layout_margin="50dp"
    android:text="Layout actividad principal" />
  <androidx.fragment.app.FragmentContainerView
    android:id="@+id/contenedor_fragmento_dinamico"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center"/>
</LinearLayout>
```

Observar que no existen los atributos **android:name y tools:layout**

Añadir un fragmento dinámicamente (3)

1. Crear un objeto de la clase del fragmento, pasándole argumentos. Alternativas:
   1.1. Como factoría. Los argumentos se pasan como parámetros y son fijos y obligatorios:

```
FragmentoEjemplo fragmentoEjemplo = FragmentoEjemplo.newInstance (param1, param2);
```

   1.2. Como instancia. Los argumentos son variables y opcionales:

```
FragmentoEjemplo fragmentoEjemplo = new FragmentoEjemplo ();
Bundle parametros = new Bundle ();
parametros.putString ("clave1", valor1); parametros.putInt ("clave2", valor2);
fragmentoEjemplo.setArguments (parametros);
```

2. Obtener la instancia del administrador de fragmentos

```
FragmentManager fragmentManager = getSupportFragmentManager ();
```

3. Crear una nueva transacción

```
FragmentTransaction transaction = fragmentManager.beginTransaction ();
```

4. Añadirlo a un *ViewGroup* del diseño de la actividad

```
transaction.add (R.id.contenedor_fragmento_dinamico, fragmentoEjemplo);
```

5. Confirmar el cambio

```
transaction.commit (); // Cuando el proceso de la IU principal pueda hacerlo
transaction.commitNow (); // Inmediatamente
```

Pasos 2 al 5 en una línea:

```
getSupportFragmentManager().beginTransaction().add(R.id.contenedor_fragmento_dinamico, fragmentoEjemplo).commit()
```

Eliminar o remplazar un fragmento dinámicamente

- Eliminar:

```
Fragment idFragmento = getSupportFragmentManager().findFragmentById(R.id.contenedor_fragmento_dinamico);
getSupportFragmentManager().beginTransaction().remove(idFragmento).commit();
```

- Remplazar por otro:

```
getSupportFragmentManager().beginTransaction().replace(R.id.contenedor_fragmento_dinamico, fragmentoNuevo).commit();
```

- Verificar si existe

```
if (getSupportFragmentManager().findFragmentById(R.id.contenedor_fragmento_dinamico)!= null
    // Ya existe
else
    // No existe. Crearlo y añadirlo
```

Fichero Java del fragmento

```
public class FragmentoEjemplo extends Fragment {
  // TODO: Rename parameter arguments, choose names that match
  // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
  private static final String ARG_PARAM1 = "param1";
  private static final String ARG_PARAM2 = "param2";
  // TODO: Rename and change types of parameters
  private String mParam1;
  private String mParam2;
  public FragmentEjemplo() {
    // Required empty public constructor
  }
  /**
   * Use this factory method to create a new instance of
   * this fragment using the provided parameters.
   *
   * @param param1 Parameter 1.
   * @param param2 Parameter 2.
   * @return A new instance of fragment FragmentoEjemplo.
   */
  // TODO: Rename and change types and number of parameters
  public static FragmentoEjemplo newInstance(String param1, String param2) {
    FragmentoEjemplo fragment = new FragmentoEjemplo ();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, param1);
    args.putString(ARG_PARAM2, param2);
    fragment.setArguments(args);
    return fragment;
  }
```

```
  @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      if (getArguments() != null) {
        mParam1 = getArguments().getString(ARG_PARAM1);
        mParam2 = getArguments().getString(ARG_PARAM2);
      }
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
      // Inflate the layout for this fragment
      return inflater.inflate(R.layout.fragment_ejemplo, container, false);
    }
  }
```

```
@Override
public void onViewCreated(View view, Bundle savedInstanceState){


}
```

Meter en este método las acciones que usan el diseño

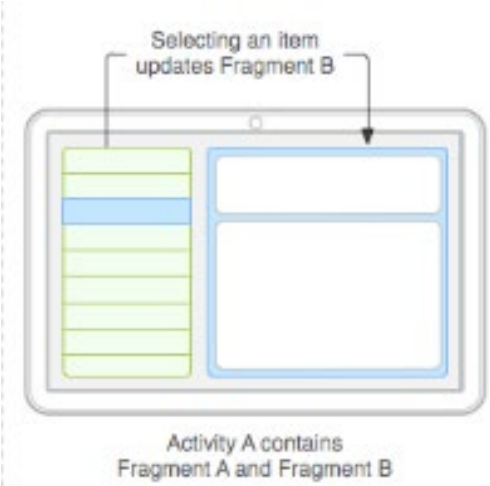
```
TextView un_textView=view.findViewById(R.id.TextView);
un_textView.setText("Parámetros recibidos: "+mParam1+",  "+mParam2);
```

Ejercicio 1

- Crea un proyecto nuevo, con las siguientes características:
  - Nombre: “7 Fragmento ejemplo”
  - Paquete: dam.fragmentoejemplo
  - Directorio: 7-FragmentoEjemplo
- El diseño de la actividad debe mostrar un fragmento añadido de forma estática y otro de forma dinámica.
- El fragmento dinámico:
  - Debe añadirse cuando se pulsa el botón, recogiendo los valores de dos `EditText` y mostrándolos
  - Debe ser eliminado cuando se vuelve a pulsar el botón

Comunicación entre fragmentos (1)

- Suponer una App con los fragmentos A y B. Cuando ocurre algo en A (por ejemplo, se selecciona un ítem), queremos que B lo sepa



- Dos formas posibles de comunicación entre A y B:
  1. El fragmento A llama directamente a un método de B:
     `fragmentoB.actualizarDato(dato);`
  2. El fragmento A se comunica a través de la actividad.

Comunicación entre fragmentos (2)

- Llamar directamente desde un fragmento a otro es una mala práctica porque:

Problema	Descripción
Acoplamiento	A necesita conocer a B, rompiendo la modularidad.
Fragilidad	Si B no está cargado aún, hay *NullPointerException*.
Poca escalabilidad	Dificulta reusar fragmentos en otras pantallas o modos (vertical/horizontal).
Complica el mantenimiento	El código se vuelve más rígido y difícil de adaptar.

Comunicación entre fragmentos (3)

- La comunicación debe hacerse a través de la actividad contenedora porque:

Ventaja	Razón
Fragmentos independientes	No se conocen entre sí
Más flexibilidad	El mismo fragmento puede usarse en múltiples actividades
Compatible con distintos layouts	Por ejemplo, modo horizontal o vertical

Comunicación entre fragmentos (4)

¿Cómo se hace esta comunicación?

- El fragmento A define una interfaz con un método al que se le pasan datos como argumentos
- La actividad implementa el método de esa interfaz
- El fragmento A invoca al método de la interfaz
- En ese método, la actividad llama al fragmento B pasándole los datos recibidos como parámetros

Ejemplo de App con comunicación entre fragmentos (1)

Una App está formada por:

- FragmentoUno: contiene un EditText y un botón
- FragmentoDos: muestra un mensaje en un TextView
- MainActivity: contiene ambos fragmentos

Objetivo: cuando se pulsa el botón de FragmentoUno, leer el contenido del EditText y mostrar ese texto en el TextView de FragmentoDos.

Ejemplo de App con comunicación entre fragmentos (2)

En FragmentoUno:

- Definir la interfaz:

```
public interface CanalComunicacion {
   void envioMensajes(String mensaje);
}
```

- Verificar que la actividad ha implementado la interfaz

```
@Override
public void onAttach(Context context) {
   super.onAttach(context);
    try {
          manejadorCanalComunicacion = (CanalComunicacion) context;
      } catch (ClassCastException e) {
          throw new ClassCastException(context.toString() + " debe implementar la interfaz CanalComunicacion");
      }
    }
```

Ejemplo de App con comunicación entre fragmentos (3)

En FragmentoUno:

- En el manejador del botón, invocar al método de la interfaz pasándole el contenido del EditText para notificárselo a la actividad:

```
@Override
    public void onClick(View view) {
        manejadorCanalComunicacion.envioMensajes(etMensaje.getText().toString());
    }
});
```

Ejemplo de App con comunicación entre fragmentos (4)

En la actividad:

- Declarar que implementa la interfaz

```
public class MainActivity extends AppCompatActivity implements FragmentoUno.CanalComunicacion {
  :::::::::
}
```

- Implementar el método del interfaz

```
@Override
public void envioMensajes(String contenidoMensaje) {
// Invocar a un método de FragmentoDos para pasarle el mensaje
FragmentoDos idFragmentoDos = (FragmentoDos)
        getSupportFragmentManager().findFragmentById(R.id.contenedorFragmentoDos);
idFragmentoDos.recibirMensajes (contenidoMensaje);  // Invocar a un método de FragmentoDos
}
```

Ejemplo de App con comunicación entre fragmentos (5)

En FragmentoDos:

- Implementar el método que invoca la actividad

```
public void recibirMensajes(String contenidoMensaje) {
   mensaje.setText("Me llama la actividad y me pasa este dato: "
         +contenidoMensaje+"\n");
}
```

Compartir datos entre la actividad y fragmentos (1)

- Se puede usar un objeto de una clase que herede de ViewModel. Ese objeto se compartirá entre la actividad y los fragmentos. Ejemplo:

```
public class DatosCompartidosViewModel extends ViewModel {
   public int unEntero;
}
```

- En la actividad:

```
private DatosCompartidosViewModel datosCompartidosViewModel;
:::::
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   datosCompartidosViewModel = new ViewModelProvider(this).get(DatosCompartidosViewModel.class);
   ::::
}
```

Compartir datos entre la actividad y fragmentos (2)

- En los fragmentos:

```
public class Fragmento extends Fragment {
    private DatosCompartidosViewModel datosCompartidosViewModel;
    ::::


    @Override
    public void onAttach(Context context) {
       super.onAttach(context);
       datosCompartidosViewModel = new
            ViewModelProvider(getActivity()).get(DatosCompartidosViewModel.class);
```

Ejercicio 2 (1)

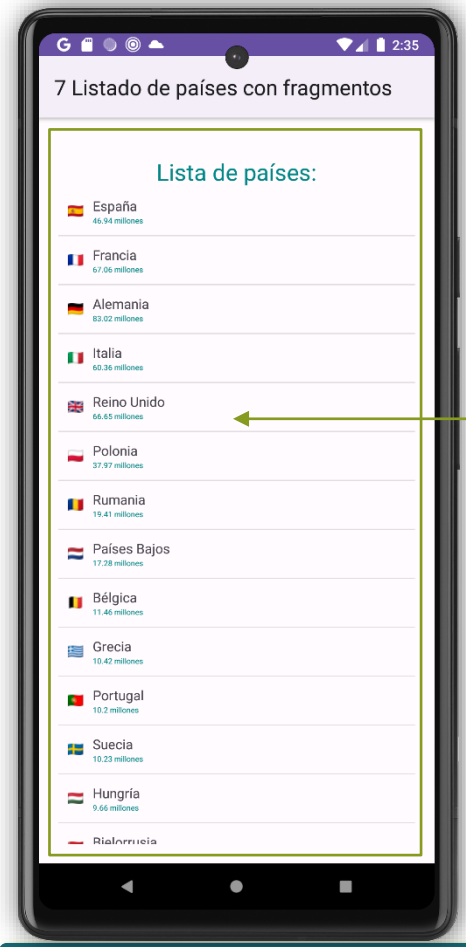
- A partir del proyecto “4 Listado países personalizado” de la sesión 4, crea un proyecto con las siguientes características:
  - Nombre: “7 Listado de países con fragmentos”
  - Paquete: dam.listadopaisesfragmentos
  - Directorio: 7-ListadoPaisesFragmentos
- Modifica la aplicación:
  - Crea dos fragmentos:
    - Fragmento listado: mostrará la lista de países
    - Fragmento detalle: mostrará la bandera, nombre de país y población
  - En la orientación vertical se usarán dos actividades:
    - Actividad inicial: mostrará el fragmento listado
    - Actividad detalle: mostrará el fragmento detalle. Se inicia al seleccionar un país por lo que recogerá los datos del país que debe suministrar la actividad inicial
  - En la orientación horizontal solo se usará la actividad principal
  - Guardar y recuperar el estado en cambios de orientación

Ejercicio 2 (2)

Actividad inicial con un diseño que incluye el fragmento listado

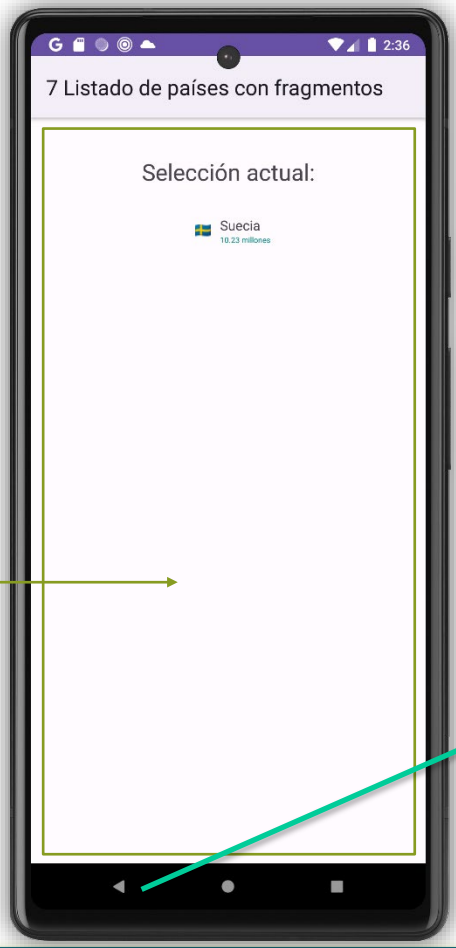
Actividad detalle con un diseño que incluye el fragmento detalle

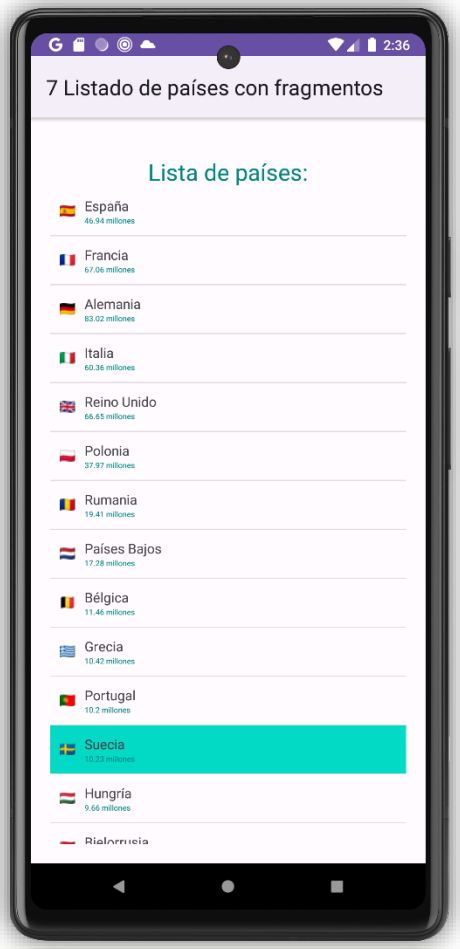
Vuelta a la actividad inicial



Fragmento listado

Fragmento detalle





Ejercicio 2 (3)

Actividad inicial con diseño de un `LinearLayout` vertical que contiene el fragmento listado, un separador y el fragmento detalle:

Fragmento listado

Fragmento detalle

Ejercicio 2 (4)

Pistas (1):

- Al pasar el código que originalmente estaba en la actividad al fragmento, algunos métodos que necesitan el objeto *context* de la actividad. Cuando el método estaba en la actividad se usaba *this*. Ahora que el método está en el fragmento hay que usar `getContext()`.

  Ejemplo:

  - En la actividad:

    ```
    Adaptador adaptadorPaises = new Adaptador(this, datosPaises.getListaPaises())
    ```

  - En el fragmento:

    ```
    Adaptador adaptadorPaises = new Adaptador(getContext(), datosPaises.getListaPaises())
    ```

  También puede ocurrir con otros métodos que estaban disponibles en la actividad, por ejemplo `getSupportFragmentManager()`, que dentro del fragmento habrá que invocarlos con `getActivity().getSupportFragmentManager()`

- Orientación horizontal: En la clase del fragmento detalle se puede crear un método para modificar el contenido de objetos de su diseño. De esta forma, cuando el fragmento detalle ya está creado, puede invocarse a ese método desde la actividad para no crear siempre un nuevo fragmento detalle cuando se desea modificar el contenido.

Ejercicio 2 (5)

Pistas (2):

- Un fragmento no tiene el método `onRestoreInstanceState()`, por lo que tendrá que recuperar los datos dentro de `onCreate()` cuando haya *Bundle*
- En el fragmento listado, tener esto en cuenta:
  - En `onCreate()` poner el código que no necesite del diseño, pues este no se ha creado todavía
  - El método `onCreateView()`, dejarlo tal y como propone la plantilla
  - Implementar `onViewCreated ()` con todas las acciones que se necesitan sobre el diseño, pues en ese punto ya está creado
- En la actividad que muestra el detalle, en la orientación vertical, dentro de `onCreate()` hay que determinar si ha habido un cambio de orientación a horizontal, pues el sistema va a reiniciar esta actividad y esto ya no tiene sentido. Por ello hay que ver la orientación actual y si es horizontal, terminar la actividad (con `finish()`)
- Tener en cuenta que, al girar el terminal, se vuelven a crear los fragmentos que existieran en la orientación anterior, además de la actividad

API Google Maps

- Permite usar mapas de *Google Maps* en las aplicaciones
- No es gratuita. Está limitada a un nº de solicitudes mensuales
- Para poderla usar en una aplicación hay que registrar la aplicación en *Google Cloud Platform Console*, asociándola a una cuenta de usuario de Google, y se obtendrá una clave que hay que introducir en la aplicación

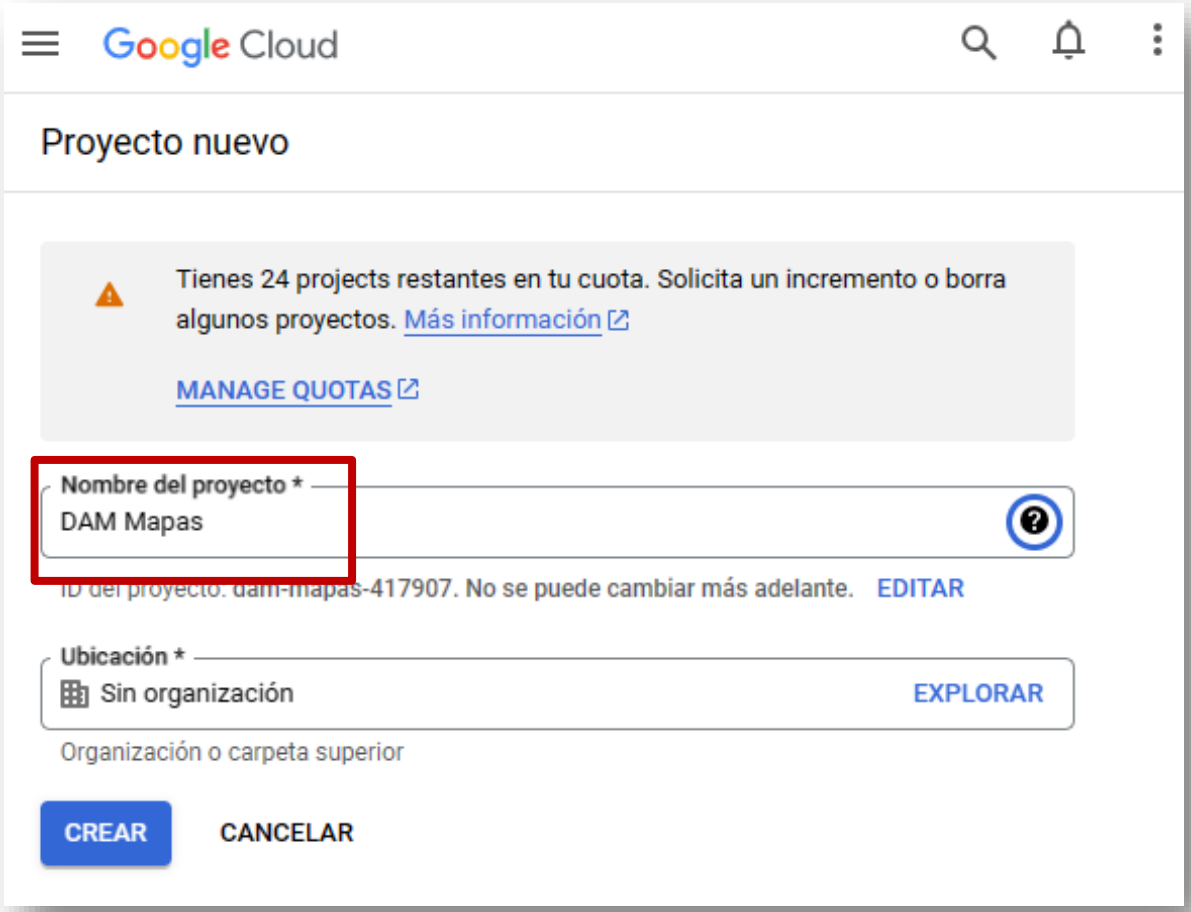
https://developers.google.com/maps/documentation/android-sdk

Crear proyecto en Google Cloud

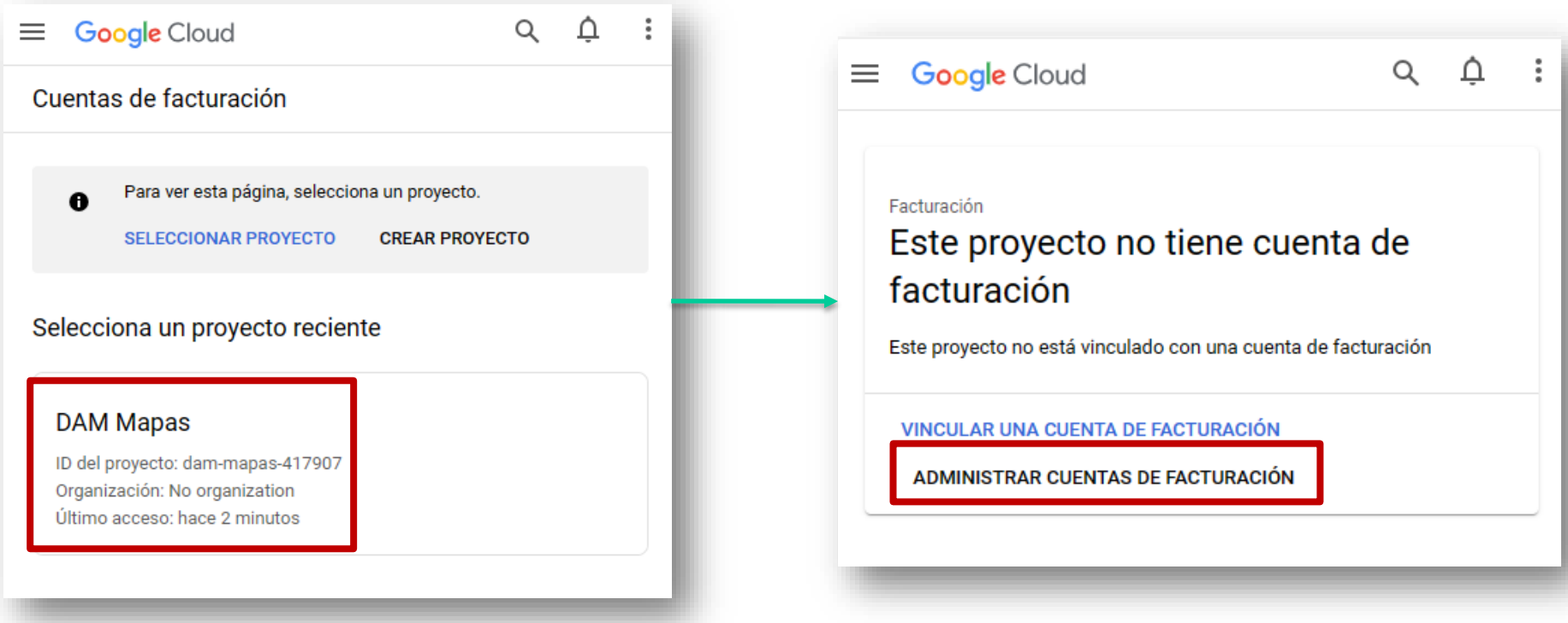
Seguir los pasos de la página: Configura tu proyecto de Google Cloud

1. Botón: *Crear un nuevo proyecto* (abre nueva ventana)
2. Autenticarse con una cuenta Google (por ejemplo la del terminal)
3. Crear proyecto nuevo
4. Habilitar la facturación

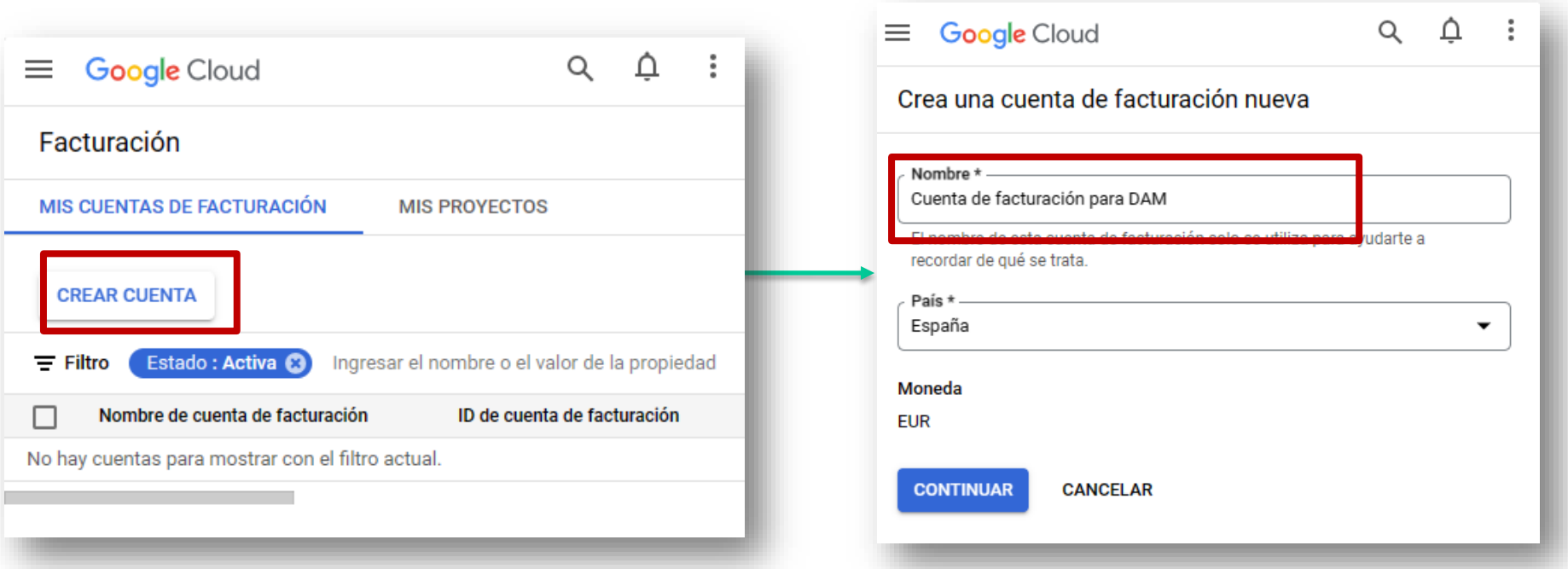
Crear proyecto nuevo en Google Cloud



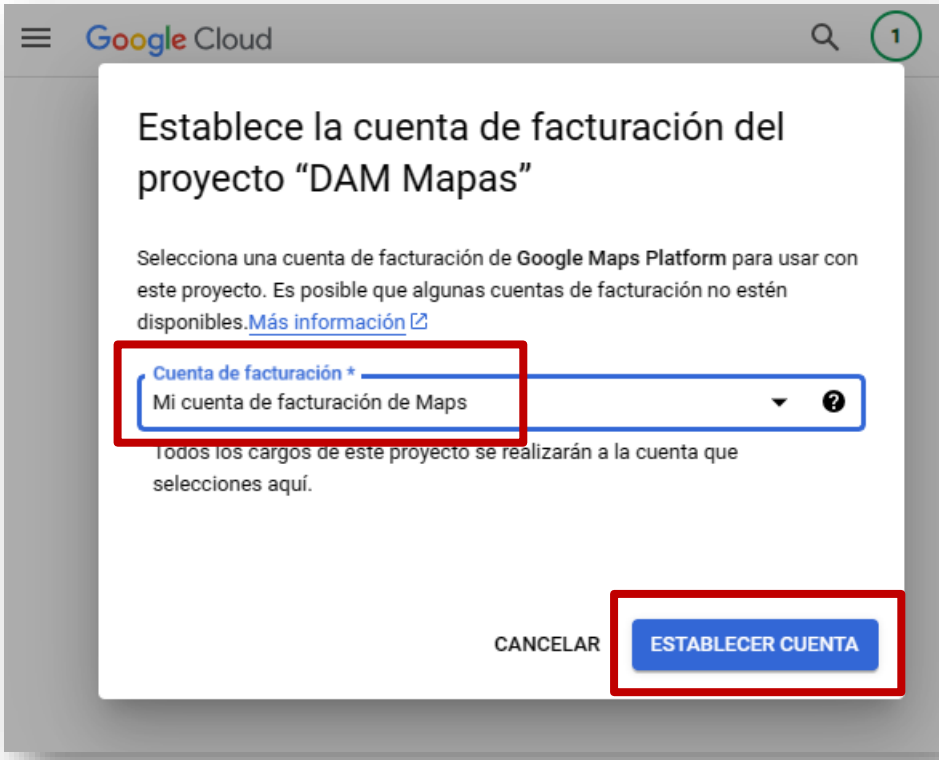
Habilitar facturación (1)



Habilitar facturación (2)



Habilitar facturación (3)

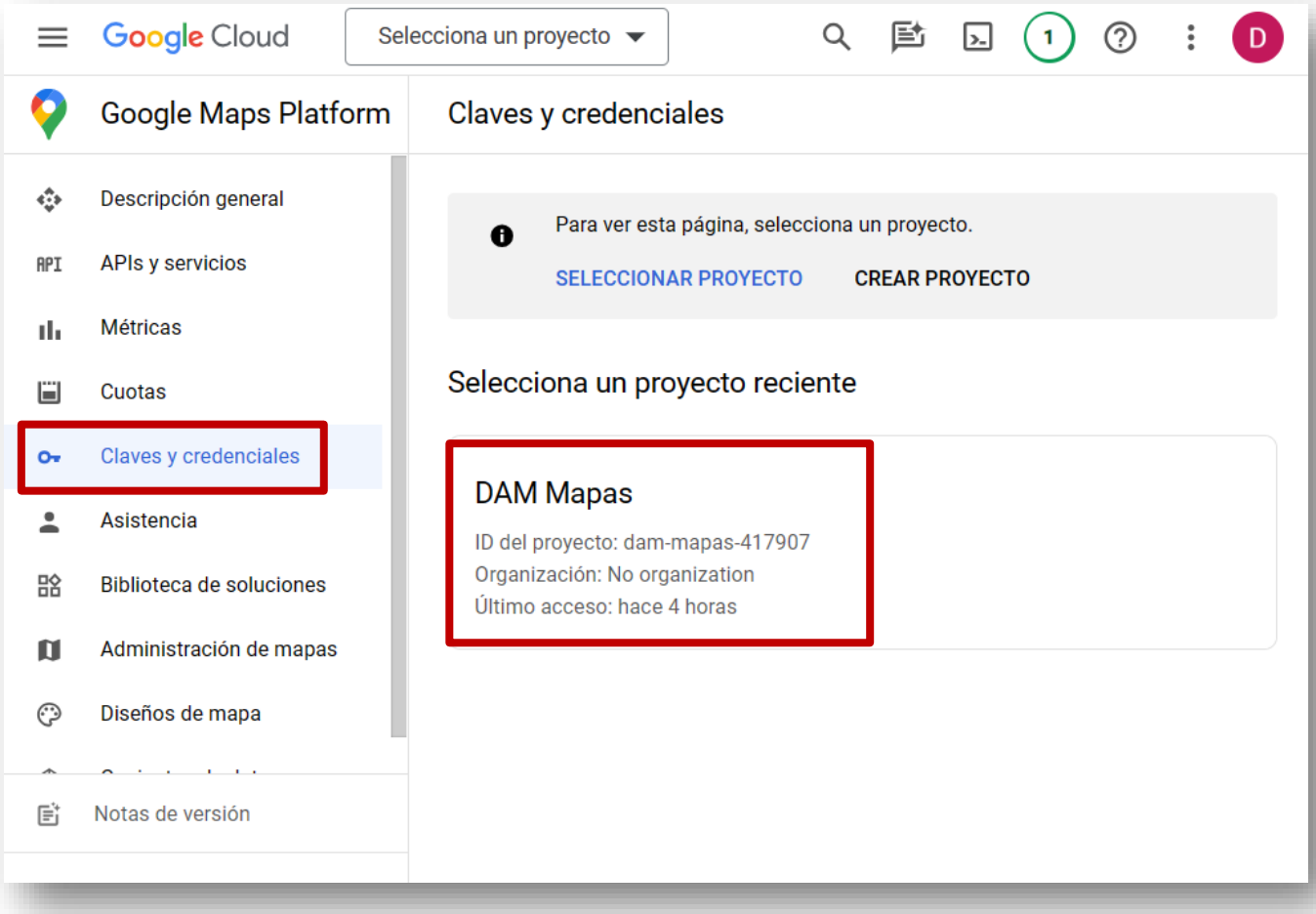


Obtener una API Key (1)

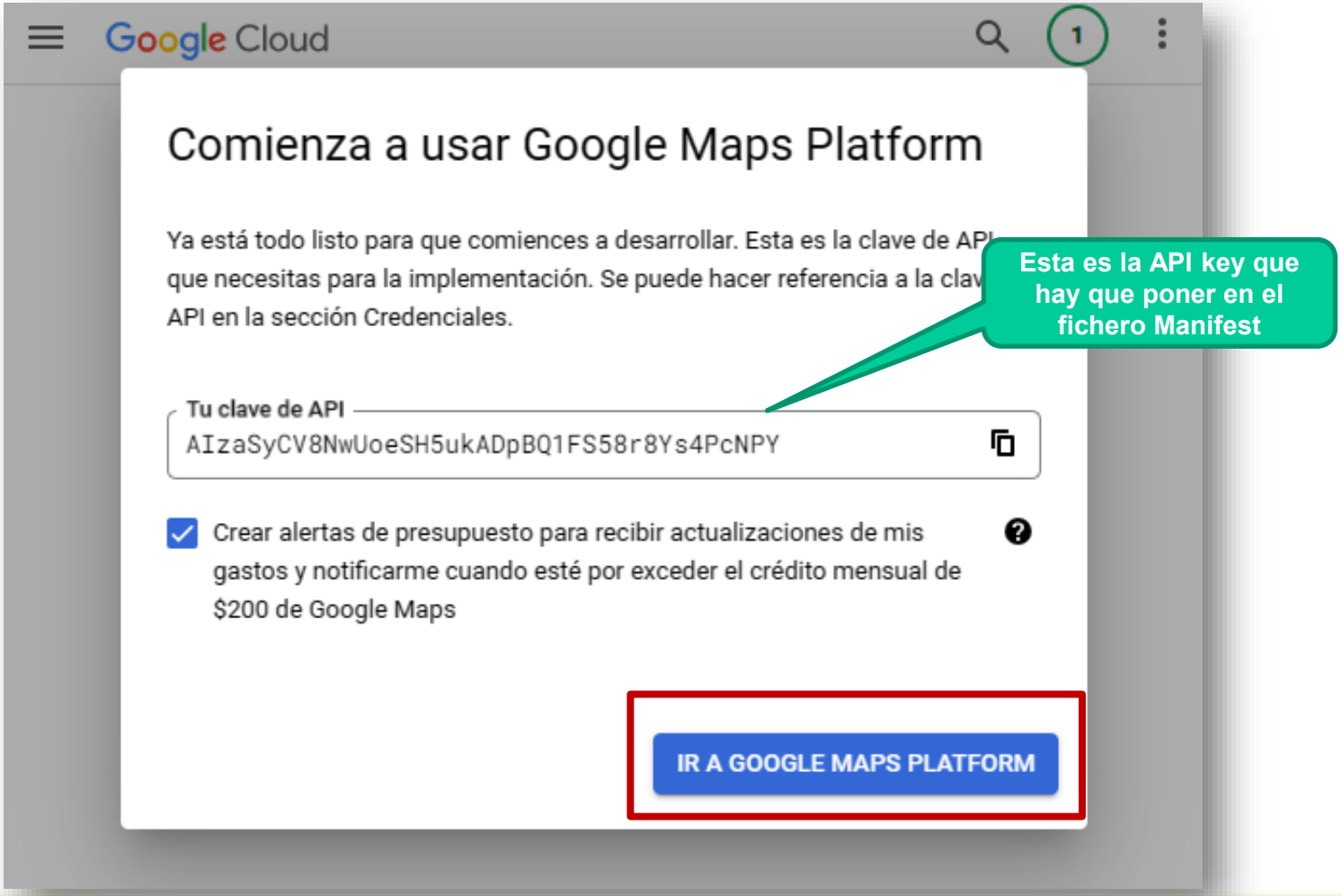
Seguir los pasos de esta página: Cómo usar claves de API

1. Botón: *Ir a la página de credenciales* (abre nueva ventana)
2. Seleccionar proyecto existente (o crear uno nuevo)
3. Obtener la API Key
4. Restringir la API Key:
   - Siempre hacerlo en aplicaciones que se compartan con otras personas
   - No hacerlo en los proyectos que subáis a Moodle para que el profesor los pueda ejecutar

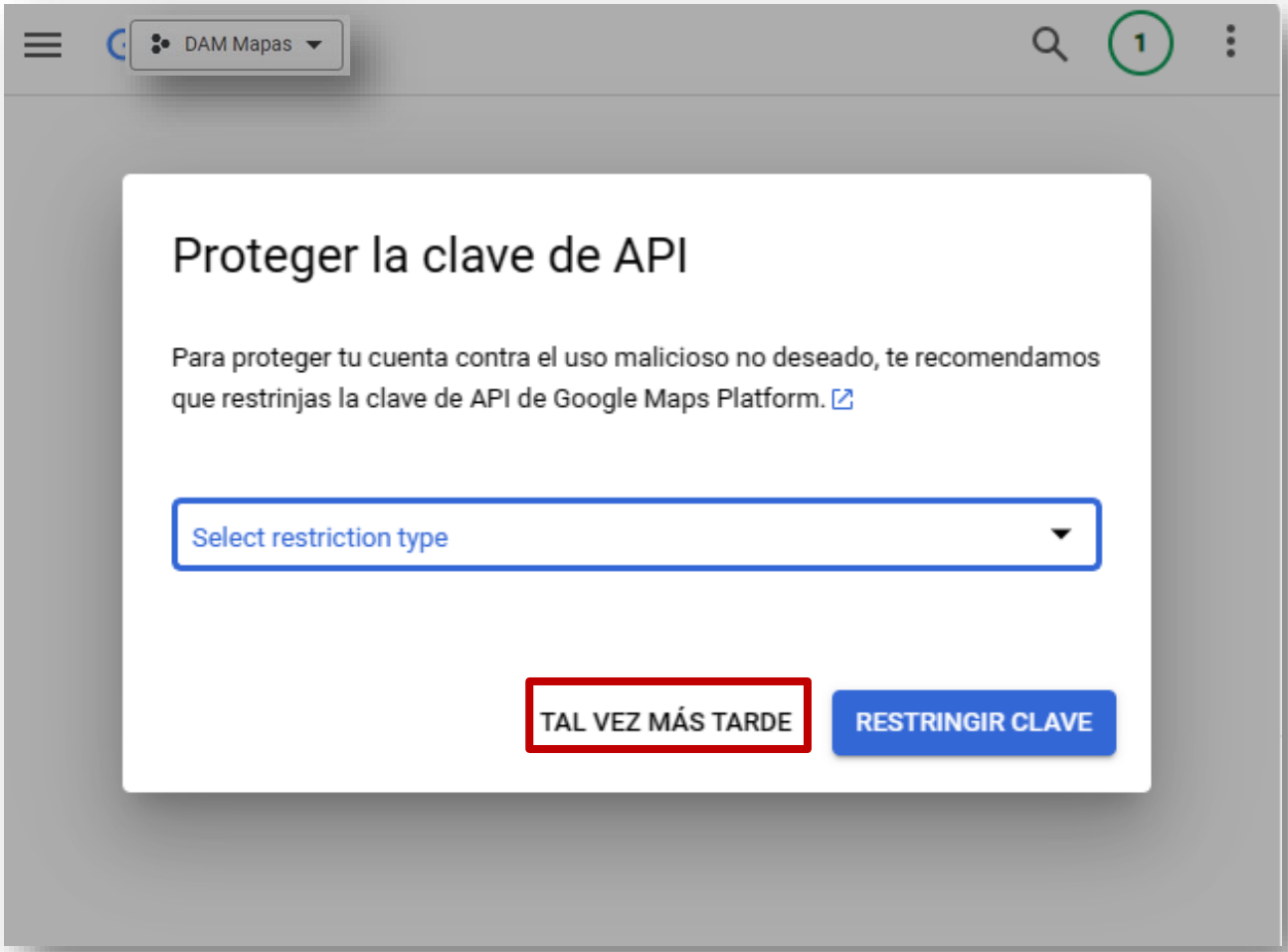
Obtener una API Key (2)



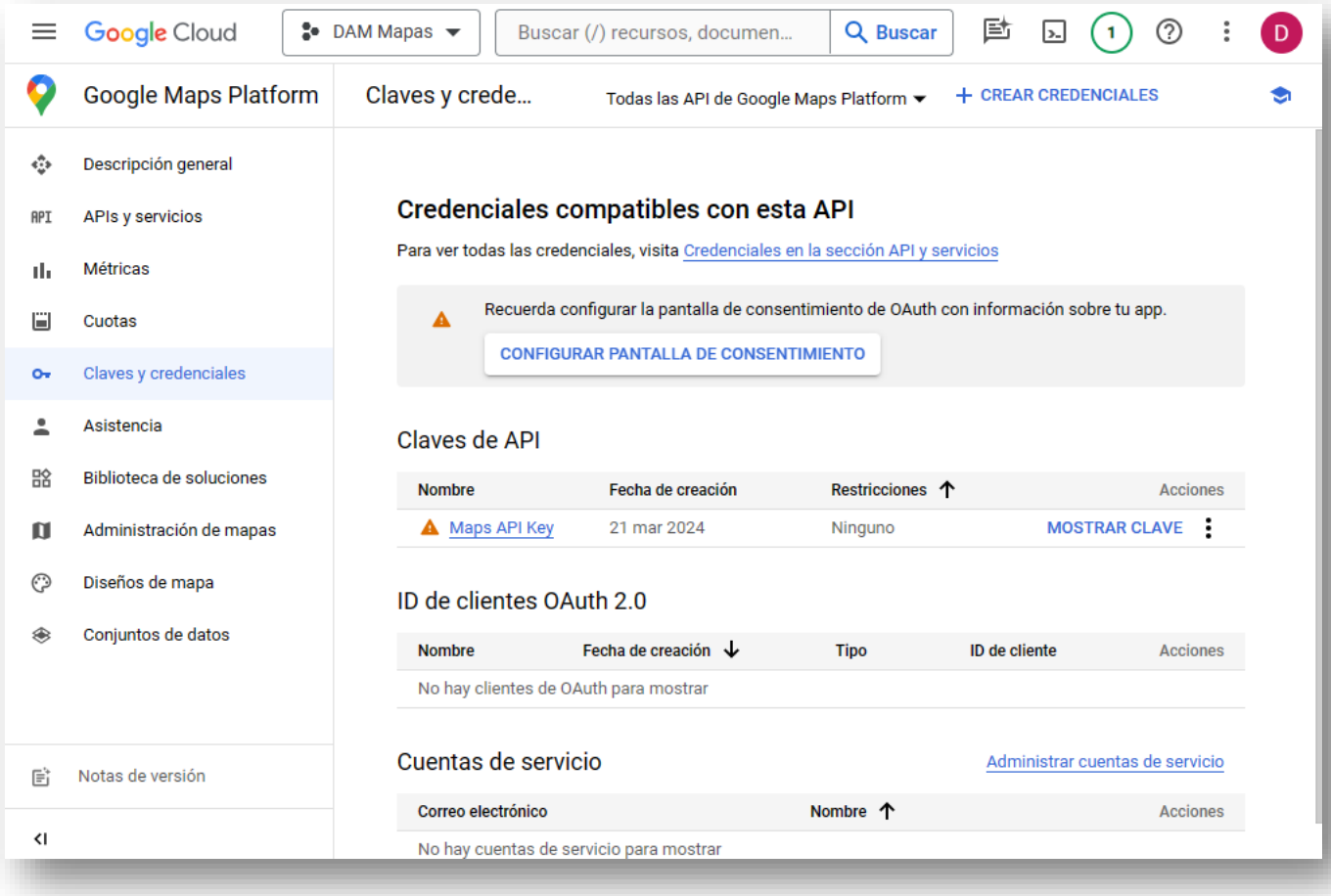
Obtener una API Key (3)



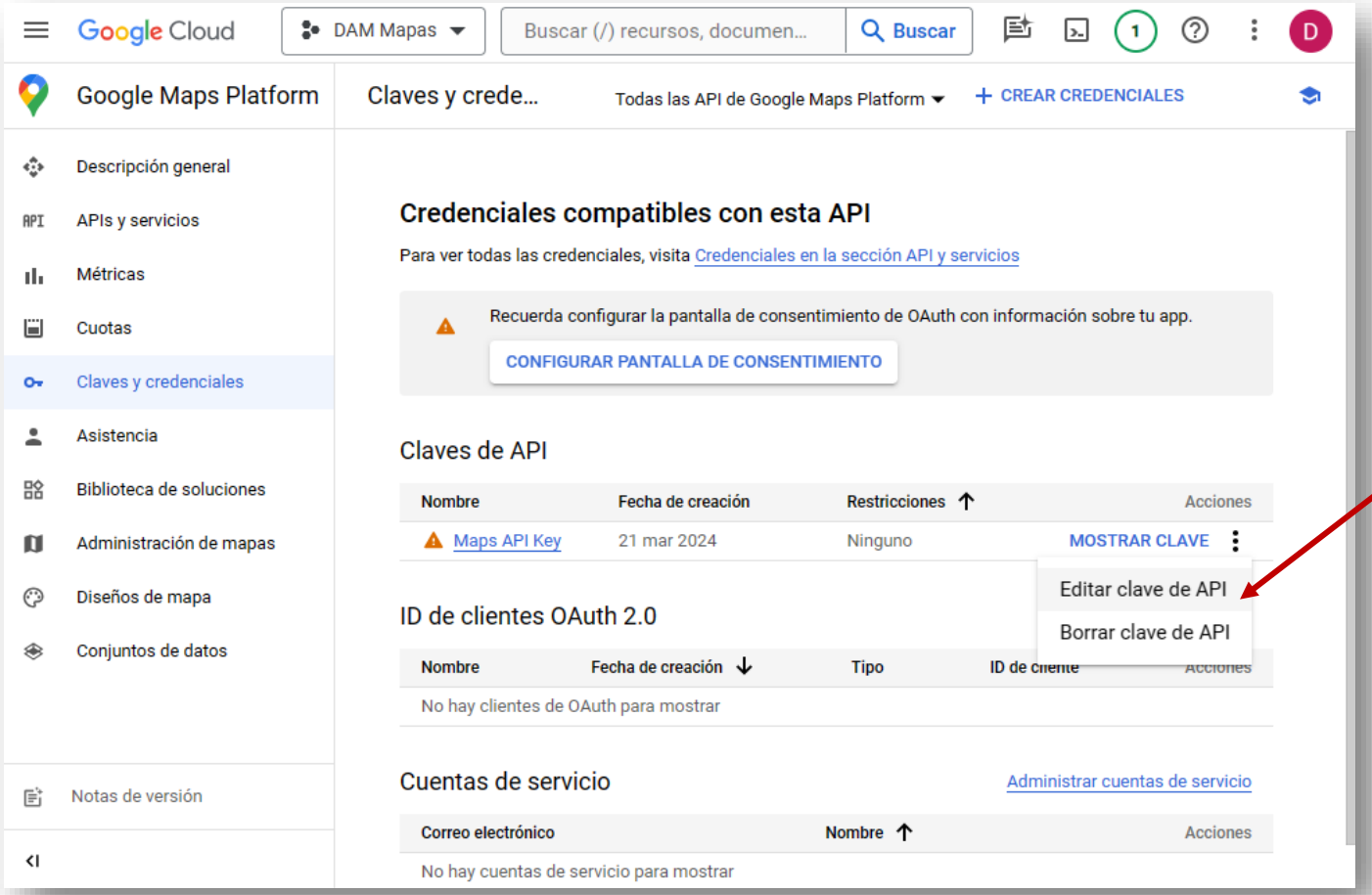
Obtener una API Key (4)



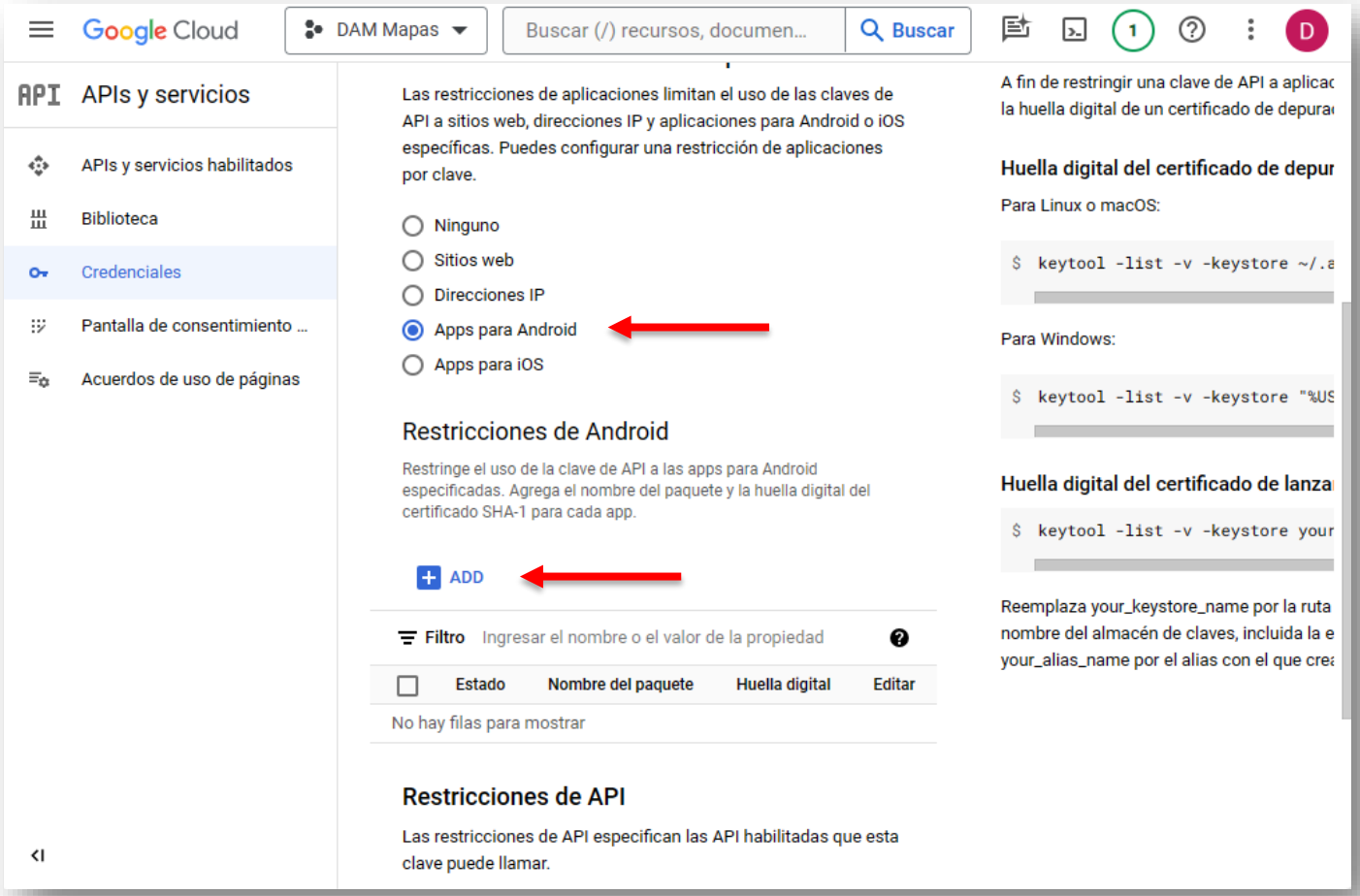
Restringir la API Key (1)



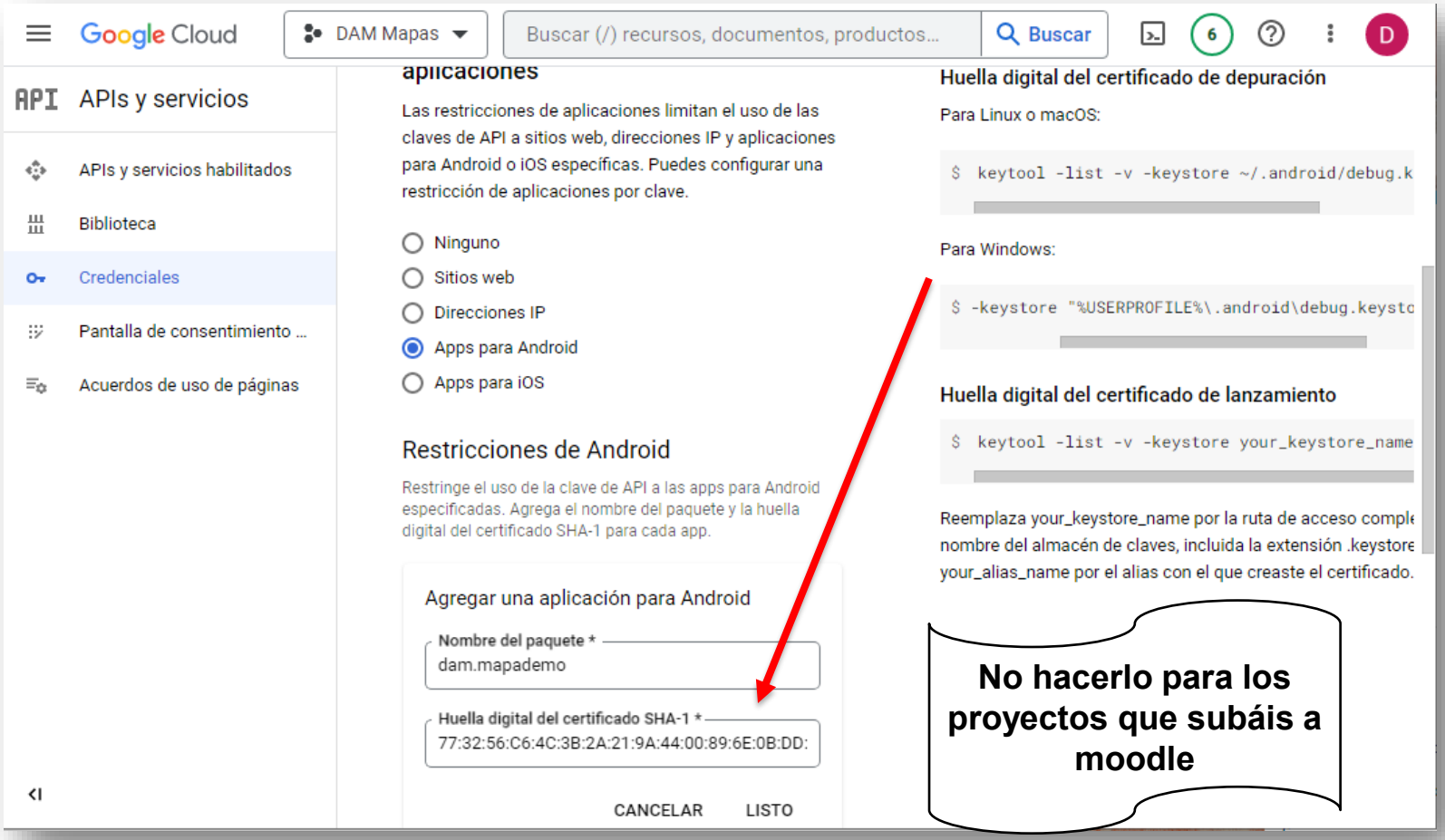
Restringir la API Key (2)



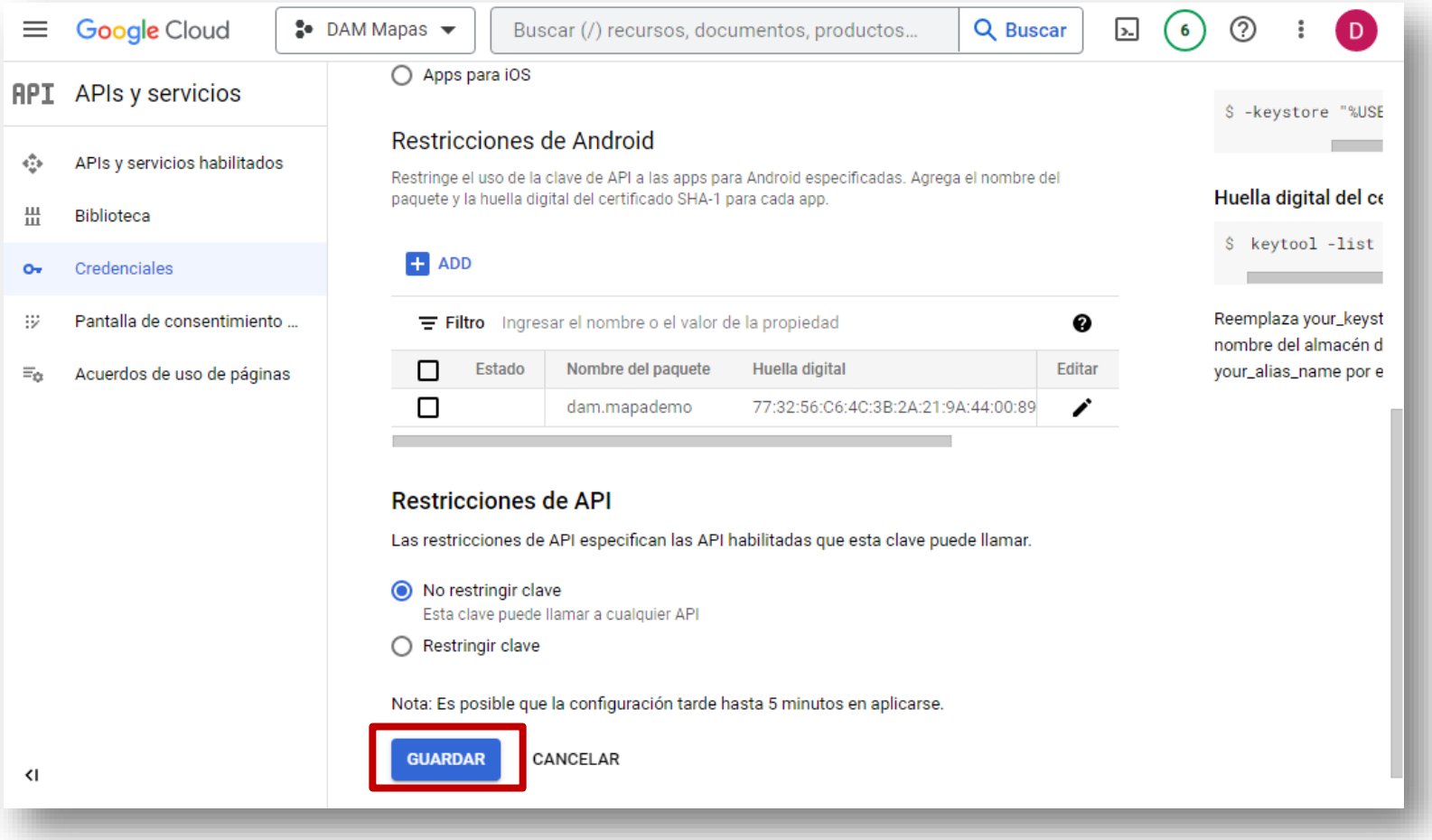
Restringir la API Key (3)



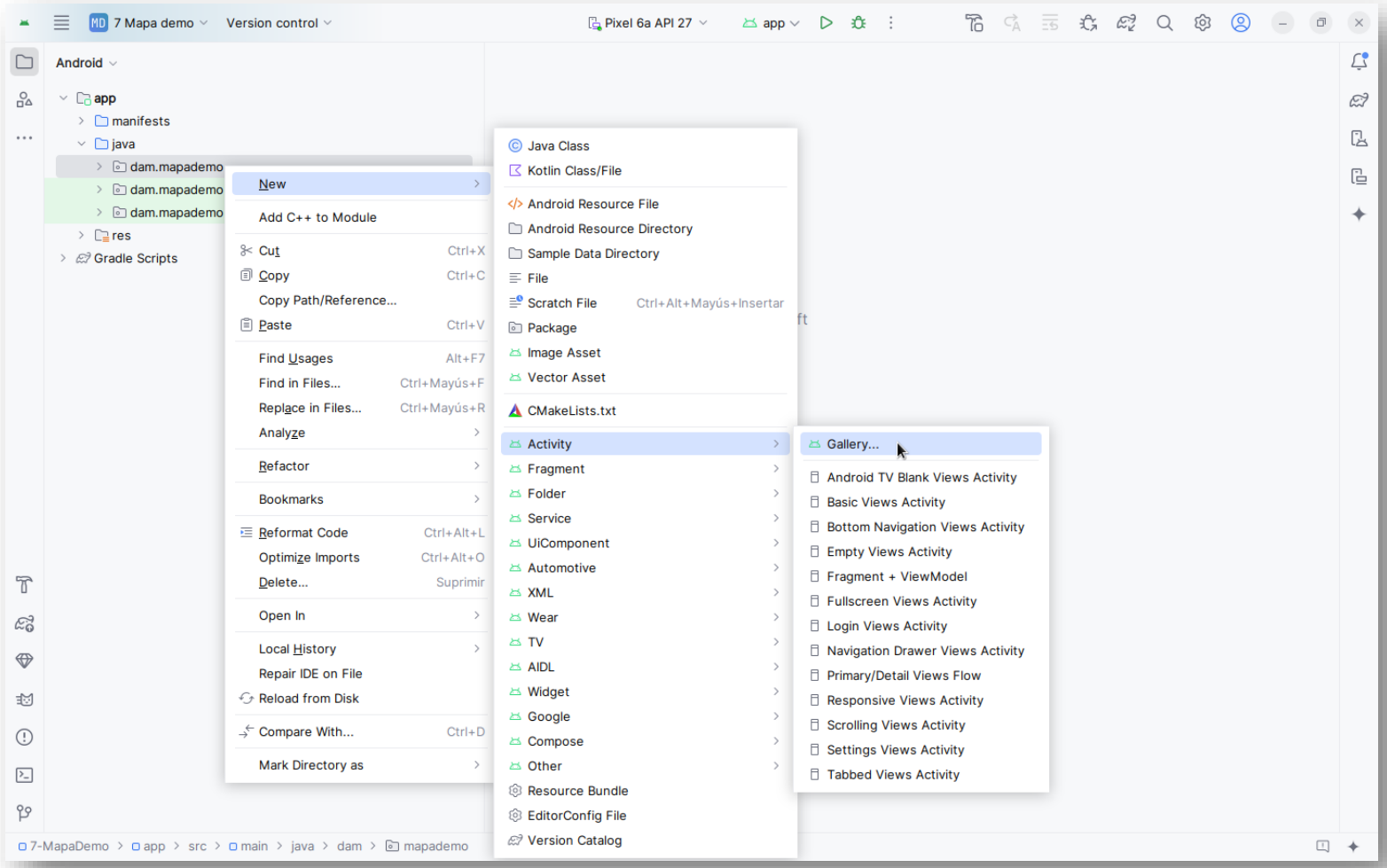
Restringir la API Key (4)



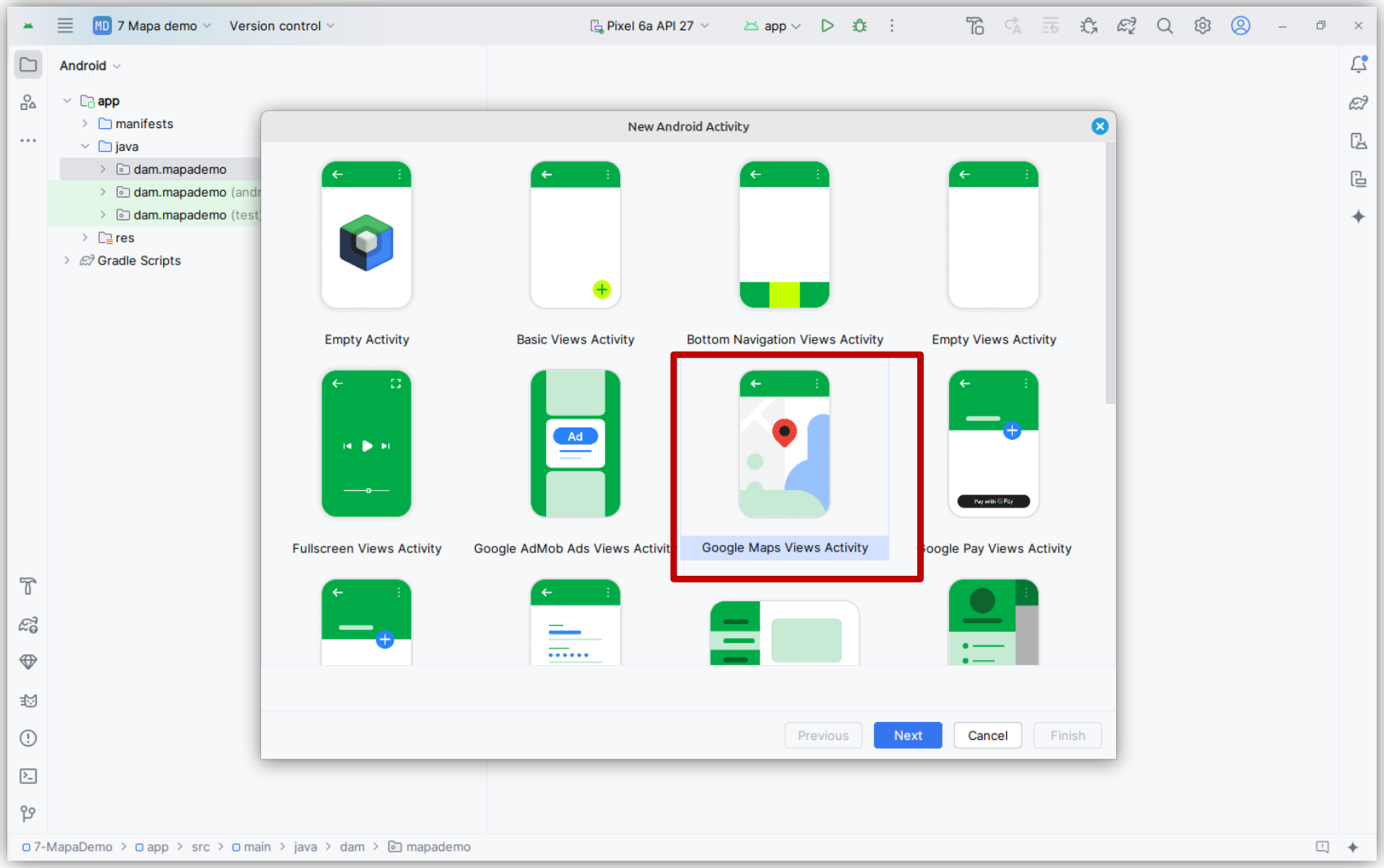
Restringir la API Key (5)



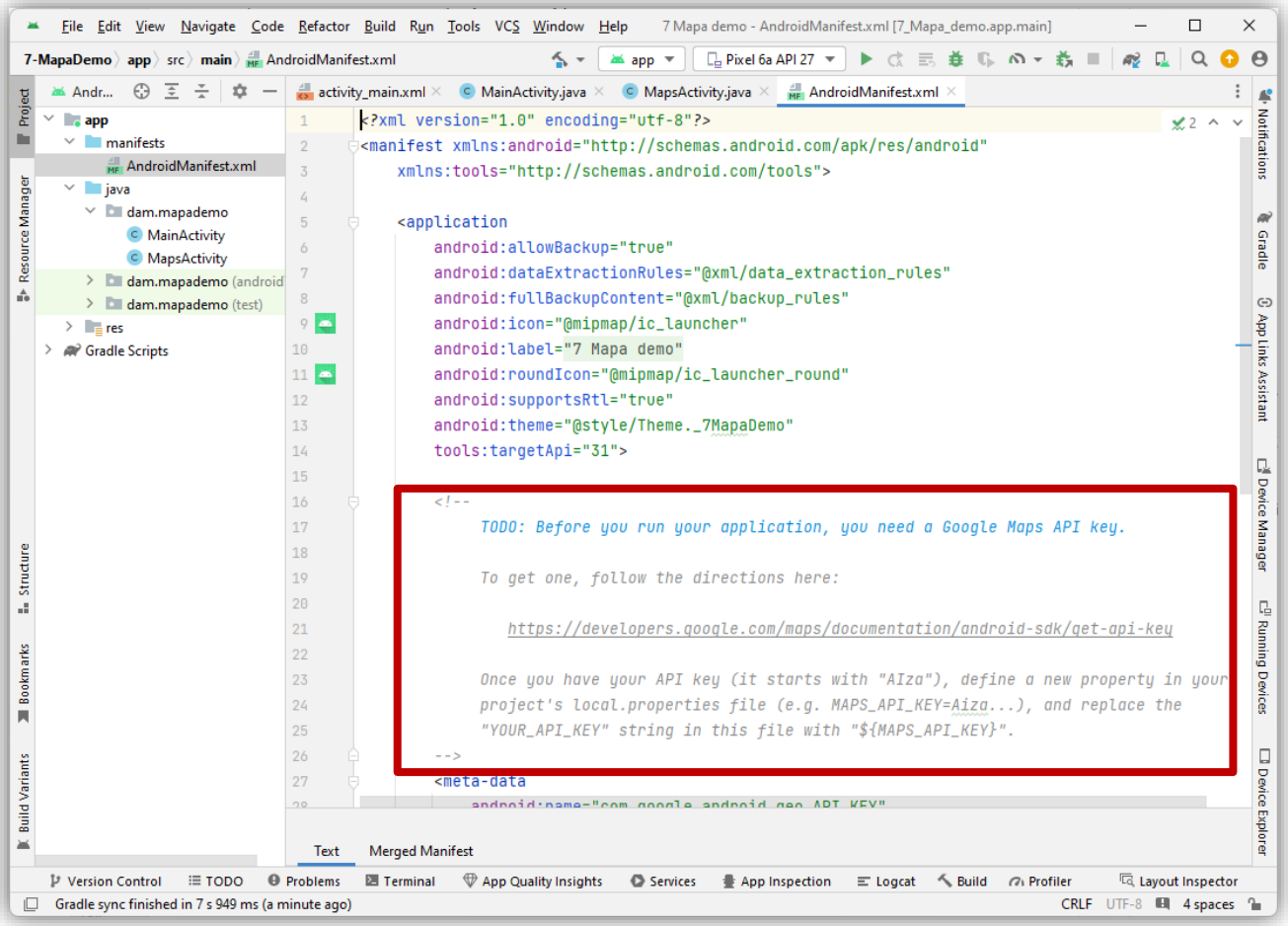
Crear proyecto con una actividad de mapa (1)



Crear proyecto con una actividad de mapa (2)



Aplicación con mapa: AndroidManifest.xml



Aplicación con mapa: AndroidManifest.xml

```
<!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow the directions here:

       https://developers.google.com/maps/documentation/android-sdk/get-api-key

    Once you have your API key (it starts with "AIza"), define a new property in your
    project's local.properties file (e.g. MAPS_API_KEY=Aiza...), and replace the
    "YOUR_API_KEY" string in this file with "${MAPS_API_KEY}".
-->
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="YOUR_API_KEY" />
```

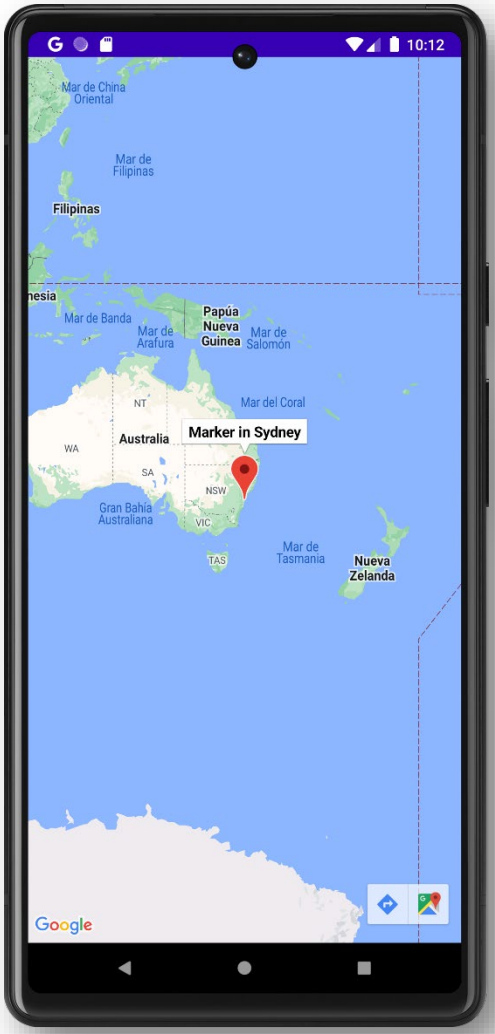
Url para obtener la Google Maps API key

Dónde poner la Google Maps API key

Actividad con mapa: diseño

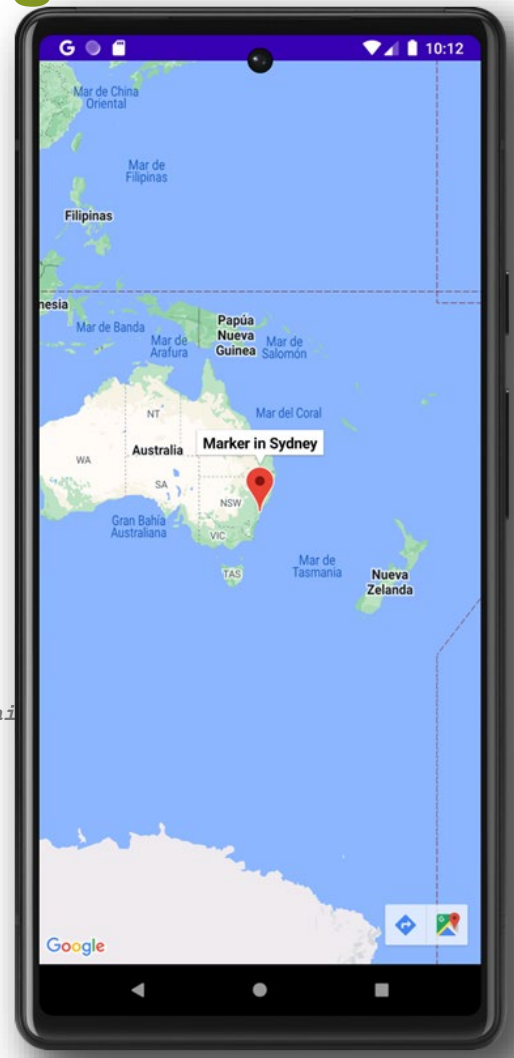
activity_maps.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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" />
```



Actividad con mapa: código

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    private GoogleMap mMap;
    private ActivityMapsBinding binding;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        binding = ActivityMapsBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        // Obtain the SupportMapFragment and get notified when the map is ready
        // to be used.
        SupportMapFragment mapFragment =
                (SupportMapFragment)getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);
    }

     /**
     * Manipulates the map once available.
     * This callback is triggered when the map is ready to be used.
     * This is where we can add markers or lines, add listeners or move the camera. In thi
     *      ...
     */
    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

        // Add a marker in Sydney and move the camera
        LatLng sydney = new LatLng(-34, 151);
        mMap.addMarker(new MarkerOptions().position(sydney).title("Marker in Sydney"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(sydney));
    }
}
```



Modificación del diseño

```
<?xml version="1.0" encoding="utf-8"?>
<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" />
```


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:text="Ejemplo"
        android:textAlignment="center"
        android:textSize="24sp" />
    <fragment
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="20"
        tools:context=".MapsActivity" />
</LinearLayout>
```

API Google Map: Tipos de vistas

- Normal:

  Valor por defecto. Mapa típico con calles y vías de comunicación

  ```
  mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
  ```

- Satélite:

  Vista de satélite. Las calles y vías de comunicación no son visibles

  ```
  mMap.setMapType(GoogleMap.MAP_TYPE_SATELLITE);
  ```

- Híbrido :

  Vista de satélite con información textual de calles y vías de comunicación

  ```
  mMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
  ```

- Terreno :

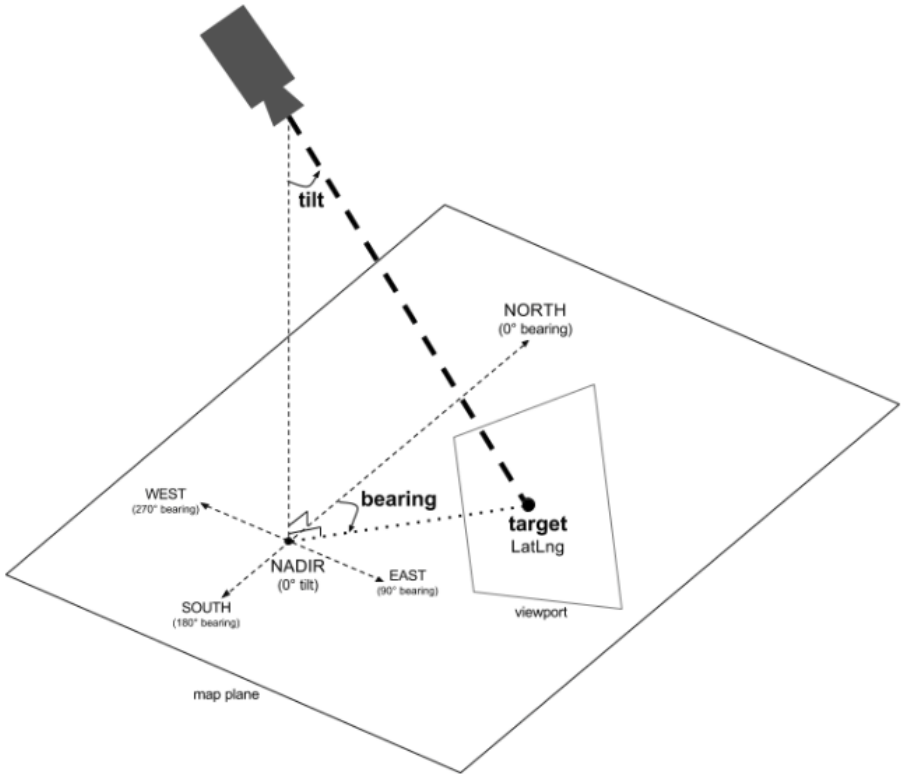
  Vista topográfica

  ```
  mMap.setMapType(GoogleMap.MAP_TYPE_TERRAIN);
  ```

Estas vistas también pueden indicarse en el atributo *map:mapType* del fichero XML. Por ejemplo `map:mapType="hybrid"`

API Google Map: posición de la cámara

La vista del mapa se modela como una cámara que mira hacia un mapa



API Google Map: posición de la cámara

- Para mover la cámara a una posición se usa la clase `CameraUpdate`
  Los objetos de esta clase se obtienen de la clase factoría `CameraUpdateFactory`
- Algunas operaciones que pueden hacerse con la cámara:
  1. Establecer una coordenada:

```
LatLng punto = new LatLng(-34, 151);
```

  2. Mover la cámara a la coordenada y hacer zoom de la vista de la cámara:
     - Inmediato :

```
mMap.moveCamera( CameraUpdateFactory.newLatLng(punto,17.0f));
```

     - Mediante una animación :

```
mMap.moveCamera( CameraUpdateFactory.newLatLng(punto));
mMap.animateCamera( CameraUpdateFactory.zoomTo(17.0f));
```

https://developers.google.com/maps/documentation/android-sdk/views

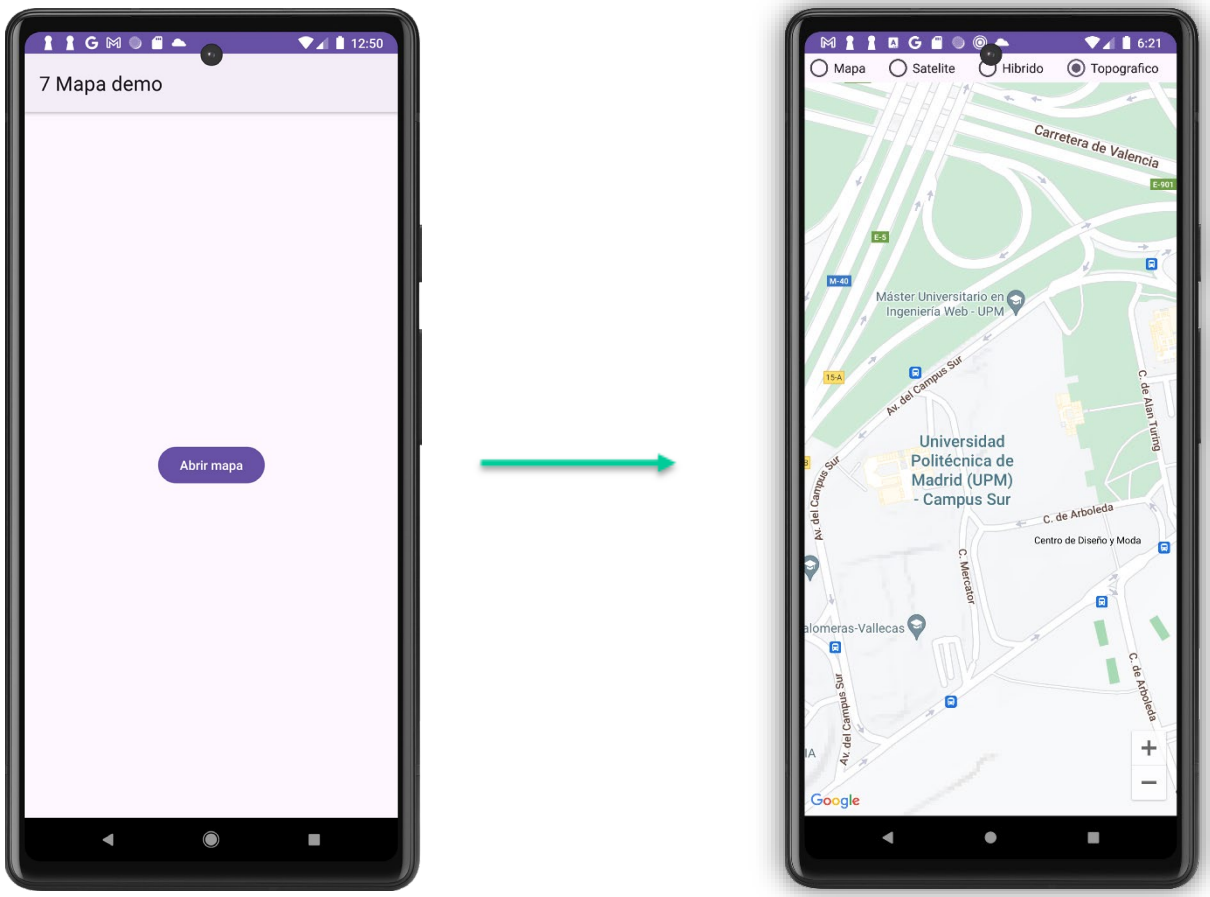
Ejercicio 3 (1)

- Crea un proyecto nuevo, con las siguientes características:
  - Nombre: “7 Mapa demo”
  - Paquete: dam.mapademo
  - Directorio: 7-MapaDemo
- Obtén un clave **google_maps_key** sin restricción
- La actividad inicial debe tener un botón para abrir la actividad del mapa
- Modifica el fichero de diseño para añadir cuatro *RadioButtons* que permitan seleccionar diferentes vistas del mapa.
- Sitúa el mapa en la localización de la ETSIST. Obtén las coordenadas de maps.google.com
- Habilita los botones de zoom: en `onMapReady()` añade:

```
UiSettings uis = mMap.getUiSettings(); uis.setZoomControlsEnabled(true);
```

- Pon un icono a la aplicación

Ejercicio 3 (2)



API Google Map: eventos del mapa (1)

- Si se quiere tratar el evento que se produce al tocar el mapa, se dispone de estas interfaces:
  - `GoogleMap.OnMapClickListener`
  - `GoogleMap.OnMapLongClickListener`
- Para capturar estos eventos hay que establecer los manejadores:
  - `GoogleMap.setOnMapClickListener(OnMapClickListener)`
  - `GoogleMap.setOnMapLongClickListener(OnMapLongClickListener)`
- Estos manejadores invocan estos métodos:
  - `onMapClick(LatLng)`
  - `onMapLongClick(LatLng)`

https://developers.google.com/maps/documentation/android-sdk/events

API Google Map: eventos del mapa (2)

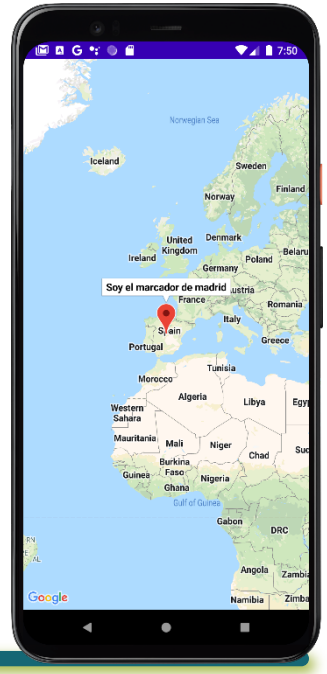
Ejemplo de implementación:

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
                                    GoogleMap.OnMapClickListener, GoogleMap.OnMapLongClickListener {

    private GoogleMap mMap;

    @Override
    public void onMapReady( GoogleMap googleMap ) {
        mMap = googleMap;

        ::::::::::::::

        // Establecer los manejadores de los de eventos al tocar el mapa:
        mMap.setOnMapClickListener( this );
        mMap.setOnMapLongClickListener( this );
    }

    @Override
    public void onMapClick( LatLng punto ) {
        Toast.makeText(this, "Has hecho clic en : "+ punto.toString(), Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onMapLongClick( LatLng punto ) {
        Toast.makeText(this, "Has hecho clic largo en : "+ punto.toString(), Toast.LENGTH_SHORT).show();
    }
}
```

API Google Map: marcadores (1)

- Un marcador es un objeto de la clase `Marker`, usado para identificar una localización concreta de un mapa.
- Se añaden con `GoogleMap.addMarker(MarkerOptions)`
- Un marcador se particulariza con un objeto de la clase `MarkerOptions`. Algunas opciones:
  - Posición en el mapa
  - Título y texto
  - El icono mostrado
  - Eventos a procesar
- Ejemplo:

```
public void onMapReady(GoogleMap mMap) {
  LatLng madrid = new LatLng(40.4284620900421, -3.70289116962);
  MarkerOptions marcadorOp=new MarkerOptions();
  marcadorOp.position(madrid);
  marcadorOp.title("Soy el marcador de madrid");
  Marker marcador = mMap.addMarker(marcadorOp);
  marcador.showInfoWindow();  // Por defecto no muestra la
                              // ventana de información
  mMap.moveCamera(CameraUpdateFactory.newLatLng(madrid));
}
```



API Google Map: marcadores (2)

- Si se quiere tratar el evento que se produce al tocar un marcador, se dispone de esta interfaz:

  `GoogleMap.OnMarkerClickListener`

- Para capturar este evento hay que establecer un manejador:

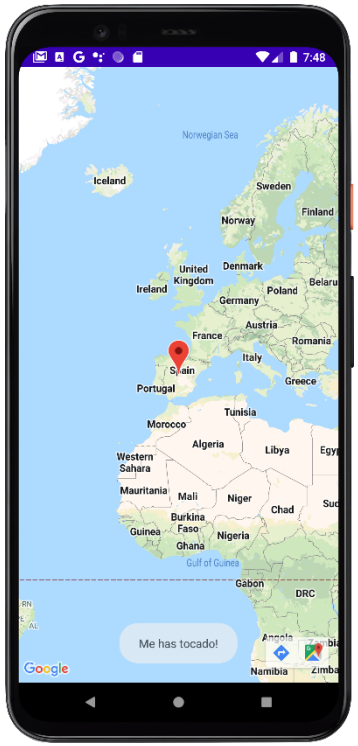
  `GoogleMap.setOnMarkerClickListener(OnMarkerClick)`

- Este manejador invoca el método `onMarkerClick(Marker)`
- El comportamiento por defecto, si no se establece manejador, es mostrar una ventana de información (con el título, por ejemplo) y centrar el mapa en el marcador.

https://developers.google.com/maps/documentation/android-sdk/marker

API GoogleMap: marcadores (3)

```
public class MapsActivity extends FragmentActivity implements OnMapReadyCallback,
                                                    GoogleMap.OnMarkerClickListener {

    private GoogleMap mMap;
        ::::::::::::

   public void onMapReady(GoogleMap mMap) {
           ::::::::::::
     LatLng madrid = new LatLng(40.4284620900421, -3.70289116962);
     mMap.addMarker(new MarkerOptions().position(madrid));
     mMap.moveCamera(CameraUpdateFactory.newLatLng(madrid))
     mMap.setOnMarkerClickListener(this); // Establecer manejador al tocar un marcador
          ::::::::::::
     }

  public boolean onMarkerClick(Marker marker) {
      Toast.makeText(this, "Me has tocado!", Toast.LENGTH_LONG).show();
      return false;
   }
}
```



Configurar los límites del mapa

- En el caso de querer que la cámara se sitúe de forma que pueda visualizar un área determinada del mapa, se dispone de la clase `LatLngBounds`, donde al instanciarla se pasan como parámetros dos objetos `LatLng()`, con las coordenadas suroeste y noreste, del área a visualizar

```
LatLng suroeste = new LatLng(40.24240793617761, -4.14428179684988);
LatLng noreste = new LatLng(40.92449288404673, -3.4521431163481346);
LatLngBounds comunidadMadrid = new LatLngBounds(suroeste,noreste);
```

- Si hubiera muchos puntos en el mapa, el área se puede calcular con la clase `LatLngBounds.Builder`

```
LatLngBounds.Builder area = new LatLngBounds.Builder();
area.include(unPunto);    // unPunto es un objeto LatLng
area.include(otroPunto);
area.include(otroPuntoMas);

LatLngBounds comunidadMadrid = area.build();  // Construye un área limitada basándose en los elementos del objeto
                                              LatLngBounds.Builder
```

Mover la cámara a un área determinada

- Para mover la cámara a un área determinada, el mapa debe estar cargado, por lo que habrá que sobrescribir el método `onMapLoaded()` de la interface `OnMapLoadedCallback` mediante el método `setOnMapLoadedCallback()`:

```
googleMap.setOnMapLoadedCallback(new GoogleMap.OnMapLoadedCallback() {
    @Override
    public void onMapLoaded() {
            mMap.animateCamera(CameraUpdateFactory.newLatLngBounds(comunidadMadrid, 10));
            o bien:
            mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(comunidadMadrid.getCenter(), 10));
    }
```

Pixel de relleno hasta el borde de la pantalla

- Otra posibilidad teniendo en cuenta el tamaño de la pantalla:

```
// Calcular el ancho y alto de la pantalla del terminal
int ancho = getResources().getDisplayMetrics().widthPixels;
int alto = getResources().getDisplayMetrics().heightPixels;
int relleno = (int) (ancho * 0.3); // Dejar libre un 30% de la pantalla

mMap.animateCamera(CameraUpdateFactory.newLatLngBounds(comunidadMadrid, ancho, alto, relleno);
```

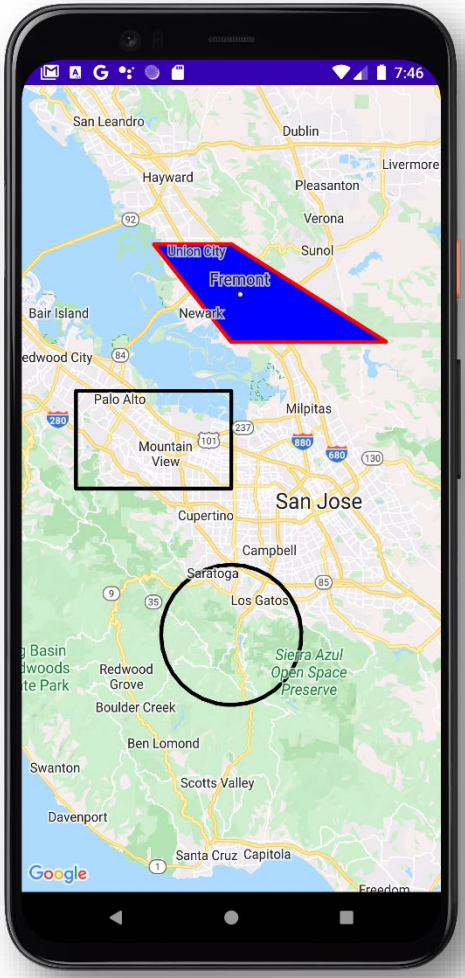
API Google Map: Formas (1)

- Clase `Polyline`:
  - Conecta, mediante líneas, un conjunto de objetos `LatLng`
  - Usa un objeto de la clase **PolylineOption** para indicar los objetos `LatLng` que se van a usar
- Clase `Polygon`:
  - Crea un polígono cerrado a partir de un conjunto de objetos `LatLng`
  - Usa un objeto de la clase **PolygonOptions** para indicar los objetos `LatLng` que se van a usar
- Clase `Circles`:
  - Dibuja un circulo centrado en un objeto `LatLng` y un radio
  - Usa un objeto de la clase **CircleOptions** para indicar la posición (objeto `LatLng`) y el radio en metros
- Es posible definir el estilo, color, … de las líneas

https://developers.google.com/maps/documentation/android-sdk/shapes

API Google Map: Formas (2)

```
LatLng punto=new LatLng(37.35, -122.0);
// Dibuja un cuadrado con Polyline
PolylineOptions polylineOptions = new PolylineOptions();
polylineOptions.add(punto);
polylineOptions.add(new LatLng(punto.latitude+0.1,punto.longitude));
polylineOptions.add(new LatLng(punto.latitude+0.1, punto.longitude-0.2));
polylineOptions.add(new LatLng(punto.latitude, punto.longitude-0.2));
polylineOptions.add(punto);
mMap.addPolyline(polylineOptions);

// Dibuja un polígono, con Polygon, con color azul de relleno y líneas rojas
PolygonOptions polygonOptions = new PolygonOptions()
        .add(new LatLng(37.50, -121.8),
             new LatLng(37.60, -122.0),
             new LatLng(37.60, -122.1),
             new LatLng(37.50, -122.0))
            .strokeColor(Color.RED)
            .fillColor(Color.BLUE);
mMap.addPolygon(polygonOptions);

// Dibuja un círculo con Circles
CircleOptions circleOptions = new CircleOptions()
            .center(new LatLng(37.20, -122.0))
            .radius(8000);
circulo=mMap.addCircle(circleOptions);
circulo.setVisible(true); // Por defecto es visible, pero se podría ocultar

// Centra el mapa en la esquina inferior derecha del cuadrado
mMap.moveCamera(CameraUpdateFactory.newLatLng(new LatLng(37.35, -122.0)));
// Establece el zoom
mMap.animateCamera( CameraUpdateFactory.zoomTo( 10.0f ) );
```



Ejercicio 4 (1)

Modifica el proyecto anterior para:

- Situar dos marcadores con su título:
  - En la entrada de la ETSIST. Este debe mostrar su título al inicio
  - En la biblioteca del campus
- Hacer zoom en un área entre esos dos marcadores
- Al tocar en algún punto, mostrar mediante un *Toast*, las coordenadas, detectando si es una pulsación corta o larga
- Al hacer clic en uno de los marcadores mostrar mediante un *Toast* un texto. Además:
  - En el de la entrada, dibujar un polígono de cuatro vértices alrededor
  - En el de la biblioteca, dibujar un círculo
  - Al hacer clic por segunda vez a un marcador, borrar el polígono o el circulo (dependiendo del marcador al que se hace clic)

Ejercicio 4 (2)