SESIÓN 8

- Estilos y temas
- Barra de herramientas
- Permisos
- Localización

Estilos

- Colección de atributos que definen el formato y apariencia que tendrá una vista:

  color, tamaño de fuente, color de fondo, …

- Ejemplo, si queremos conseguir esto:

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Sin estilo"
   android:textColor="#4164E8"
   android:typeface="sans" />
```

https://developer.android.com/guide/topics/ui/look-and-feel/themes

Estilos

Lo podemos conseguir definiendo un estilo y aplicándolo

res/values/styles.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="UnEstilo“
                  parent="TextAppearance.AppCompat">
      <item name="android:textColor">#4164E8</item>
      <item name="android:typeface">sans</item>
   </style>
</resources>
```

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Con estilo"
   style="@style/UnEstilo"/>
```

Herencia de estilos

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


   <style name="UnEstilo"
                  parent="TextAppearance.AppCompat">
      <item name="android:textColor">#4164E8</item>
      <item name="android:typeface">sans</item>
   </style>


   <style name="UnEstilo.Grande">
      <item name="android:textSize">24pt</item>
   </style>


   <style name="UnEstilo.Grande.Negrita">
      <item name="android:textStyle">bold</item>
   </style>


</resources>
```

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Con estilo"
   style="@style/UnEstilo"/>


<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Con estilo grande"
   style="@style/UnEstilo.Grande"/>


<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Con estilo grade y negrita"
   style="@style/UnEstilo.Grande.Negrita"/>
```

Tema

- Agrupación, con nombre, de estilos aplicados a toda la actividad o aplicación. Ej. “DarkTheme”, “LightTheme”, ...
- Define estilos con nombres semánticos, como “colorPrimary”, “colorSecondary”, “statusBarColor”, …
- Se define en el fichero de manifiesto

Aplicar un tema a toda una aplicación:

```
<application android:theme="@style/UnTema">
```

Aplicar un tema a una actividad en concreto:

```
<activity android:theme="@style/OtroTema">
```

Lista de todos los estilos y temas disponibles en Android en:
http://developer.android.com/reference/android/R.style.html

Tema por defecto

res/values/themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
   <!– Tema básico de la aplicación: -->
   <style name="Base.Theme._NombreAplicación" parent="Theme.Material3.DayNight.NoActionBar">
      <!– Redefinición de los colores del tema: -->
      <!– Color primario de la aplicación: -->
      <item name="colorPrimary">@color/purple_500</item>
      <item name="colorPrimaryVariant">@color/purple_700</item>
      <item name="colorOnPrimary">@color/white</item>
      <!-- Color primario de la aplicación: -->
      <item name="colorSecondary">@color/teal_200</item>
      <item name="colorSecondaryVariant">@color/teal_700</item>
      <item name="colorOnSecondary">@color/black</item>
      <!– Color de la barra de estado: -->
      <item name="android:statusBarColor" tools:targetApi="l">?attr/colorPrimaryVariant</item>
      <!– Color de texto: -->
      <item name="android:textColor">@color/text</item>
   </style>

   <style name="Theme._NombreAplicación" parent="Base.Theme._NombreAplicación" />
</resources>
```

res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="purple_200">#FFBB86FC</color>
   <color name="purple_500">#FF6200EE</color>
   <color name="purple_700">#FF3700B3</color>
   <color name="teal_200">#FF03DAC5</color>
   <color name="teal_700">#FF018786</color>
   <color name="black">#FF000000</color>
   <color name="white">#FFFFFFFF</color>
   <color name=“text">#FF000000</color>
</resources>
```

Colores de temas

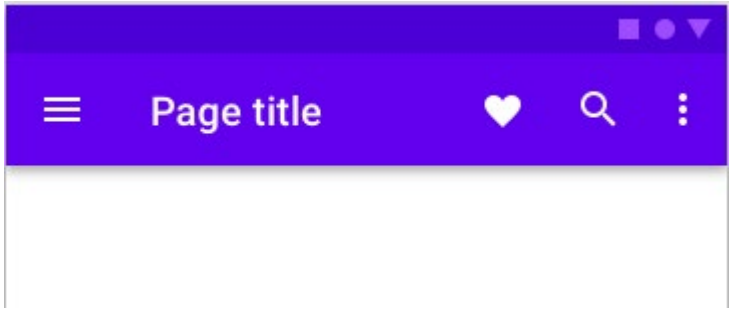
https://m3.material.io

Barra de herramientas (1)

- Existen varios tipos de barras de herramientas que pueden mostrarse en la parte superior de la GUI: *Toolbar*, *ActionBar*, *TabBar*, etc.
- En un proyecto de Android Studio con la plantilla *Empty Views Activity* se usa un tema sin *ActionBar*: *"Theme.Material3.Light.NoActionBar"*
- Si se quiere usar una *ActionBar* se puede usar un tema que los tenga: *"Theme.Material3.Light"*
- Un proyecto de Android Studio que use un tema con *ActionBar*:
  - Muestra en la parte superior el nombre de la aplicación definido en el fichero de manifiesto (atributo `android:label` del elemento `application`)
  - Cada actividad puede personalizar el título en el fichero de manifiesto (atributo `android:label` del elemento `activity`)

Barra de herramientas (2)

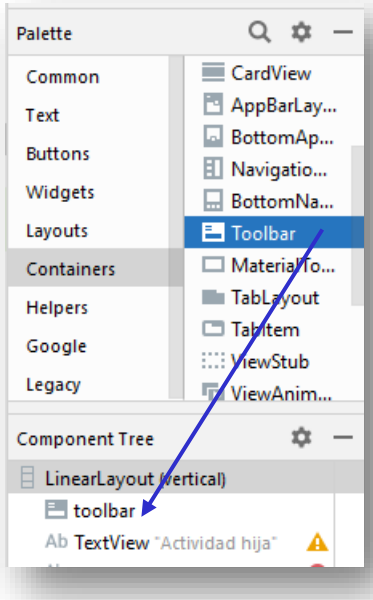
- La *ActionBar* es muy limitada, por lo que puede substituirse por la *Toolbar* que permite mas personalización. Por ejemplo, permite indicar en el diseño donde se mostrará, su tamaño, contenido, …
- Pero la *Toolbar*, al ser mas personalizable, obliga a generar más código para lo que antes se hacía de forma automática con la *ActionBar*. Una alternativa es usar un wrapper de la *Toolbar* para usarla como *ActionBar*



- Referencias:
  - https://m3.material.io/components/top-app-bar/overview
  - https://developer.android.com/training/appbar
  - https://developer.android.com/reference/androidx/appcompat/widget/Toolbar

Agregar Toolbar en el diseño de la actividad

```
<LinearLayout ..... >
        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="?attr/colorPrimary"
            android:minHeight="?attr/actionBarSize"
            app:theme="@style/actionBarTheme"
            app:titleTextColor="?attr/colorOnPrimary" />
    <Otros elementos/>
</LinearLayout>
```



Mostrar un título en la Toolbar

Mediante el método `setTitle()` de la *Toolbar*. Ejemplo:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Toolbar barraHerramientas = findViewById(R.id.toolbar);
    barraHerramientas.setTitle(getString(R.string.app_name));

}
```

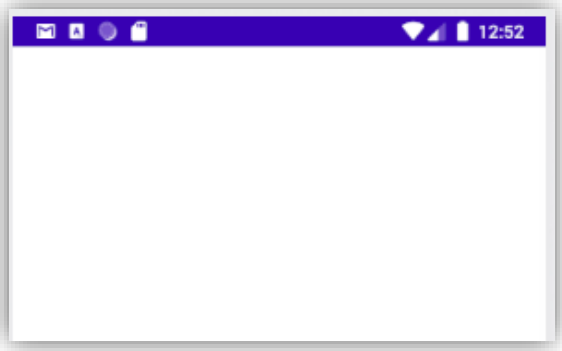
Usar la Toolbar como una ActionBar

- En el método `onCreate()` de la actividad, se debe indicar que se va a usar la *Toolbar* como una *ActionBar* mediante: `setSupportActionBar()`

```
import androidx.appcompat.widget.Toolbar;
:::
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Toolbar barraHerramientas = findViewById(R.id.toolbar);
    setSupportActionBar(barraHerramientas);
}
```

- Posteriormente se puede ocultar/mostrar

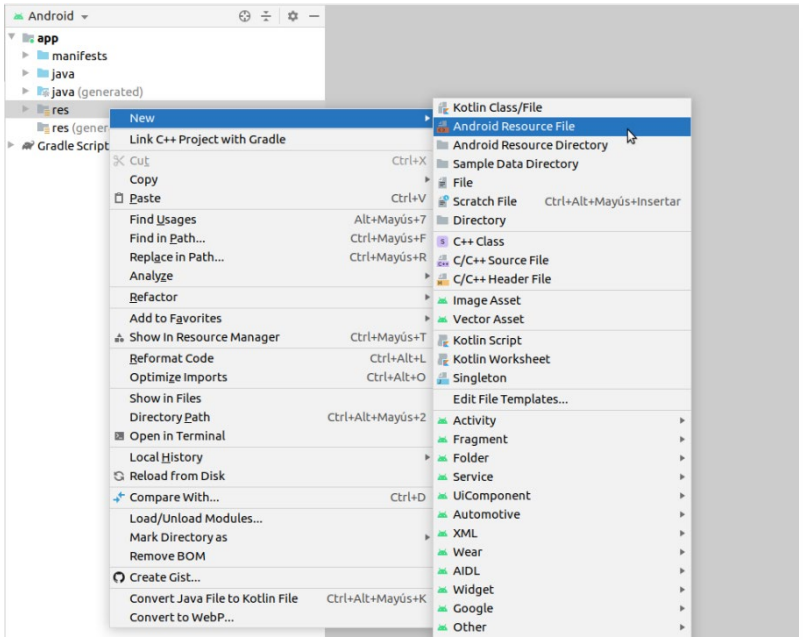
```
import androidx.appcompat.app.ActionBar;
:::
ActionBar actionBar = getSupportActionBar() ;
actionBar.hide();
actionBar.show();
```

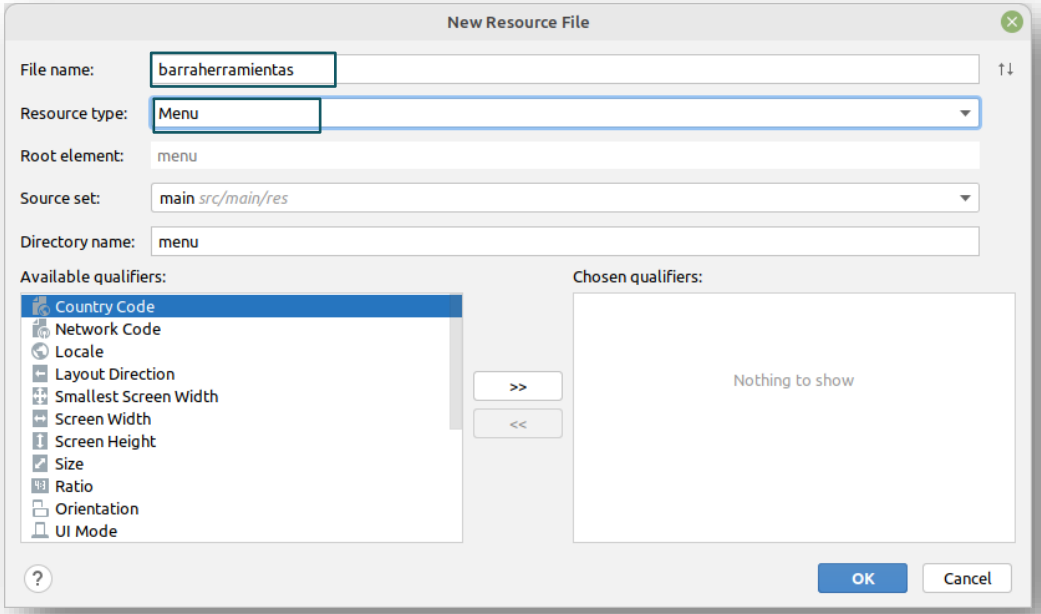


https://developer.android.com/reference/android/app/ActionBar

Agregar botones y menú

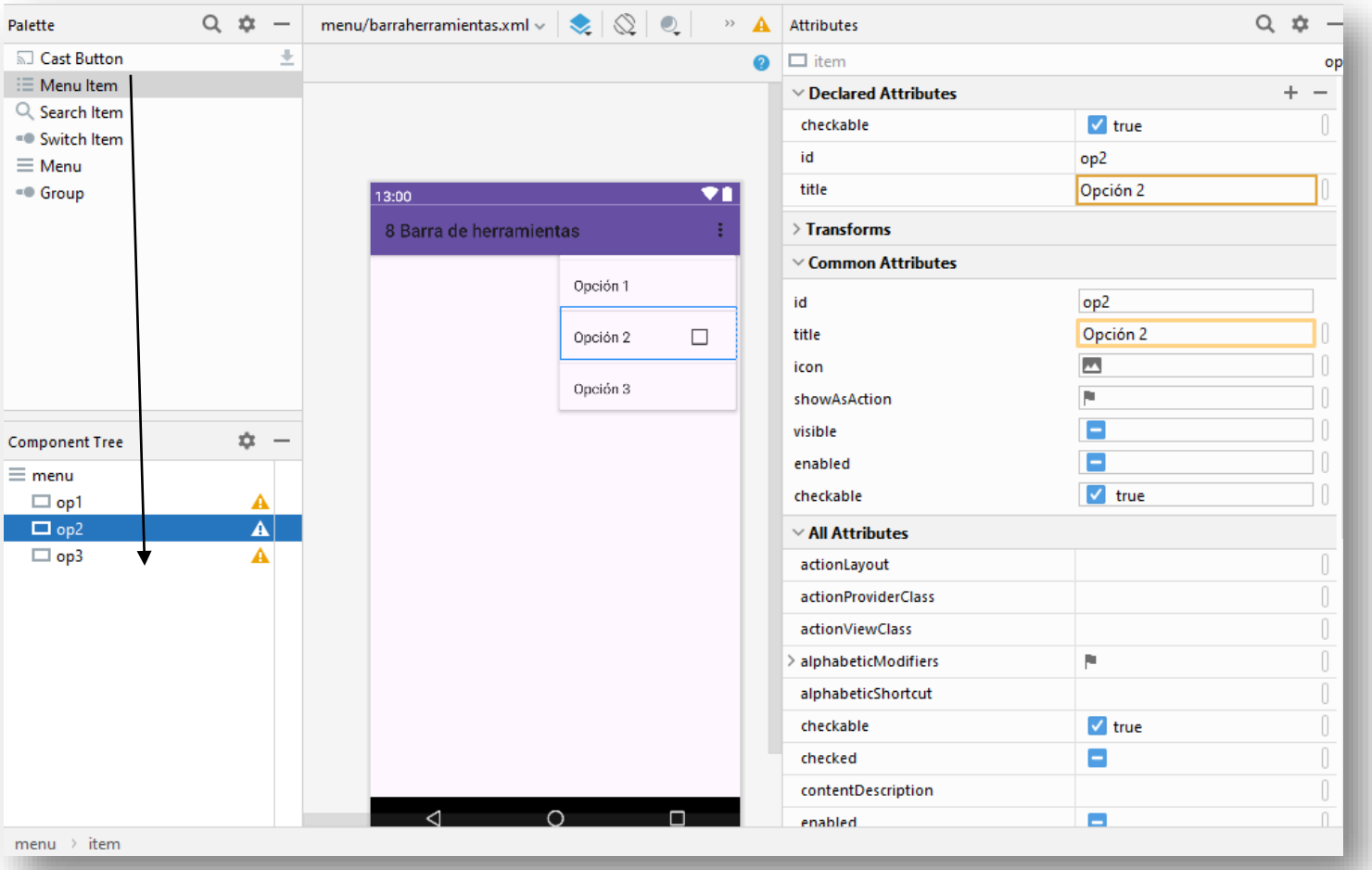
- En la barra de herramientas se pueden agregar botones (texto o iconos) y menús desplegables a izquierda y derecha
- Se necesita crear un recurso de menú XML:





https://developer.android.com/guide/topics/ui/menus

Agregar botones de acción



XML del recurso de menú

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/op1"
        android:icon="?attr/actionModeShareDrawable"
        android:title="Opción 1"
        app:showAsAction="ifRoom"
        app:iconTint="?attr/colorOnPrimary" />
    <item
        android:id="@+id/op2"
        android:checkable="true"
        android:title="Opción 2" />
    <item
        android:id="@+id/op3"
        android:title="Opción 3" />
    <group android:checkableBehavior="single" >
        <item
            android:id="@+id/op4"
            android:title="Opción 4" />
        <item
            android:id="@+id/op5"
            android:title="Opción 5" />
    </group>
</menu>
```

Icono a mostrar (opcional)

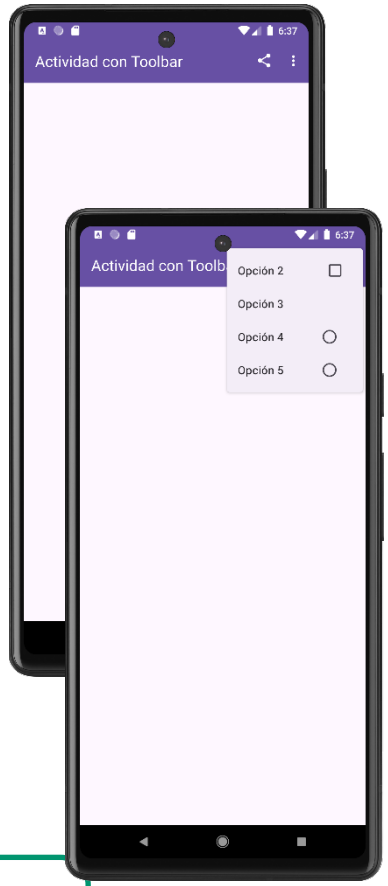
Texto a mostrar

Opcional: mostrar en la barra si hay espacio. Otra opción es *always* o *never*

Opcional: color del icono (si se muestra)

Seleccionable (opcional)

Los ítems del grupo se comportarán como *radio button*



Hacer visible el recurso del menú

En la actividad es necesario sobrescribir el método `onCreateOptionsMenu()`

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.barraherramientas, menu);

    // En el caso de que una opción fuera seleccionable, se
    // podría dar un valor inicial
    menu.findItem(R.id.op2).setChecked(true);
    return true;
}
```

Cambiar el color del icono que da acceso al menú

El color del icono por defecto es negro. Se puede cambiar a blanco con el método `setTint()`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  :::
  Toolbar barraHerramientas = findViewById(R.id.toolbar);
  Drawable iconoMenu = barraHerramientas.getOverflowIcon();
  if (iconoMenu != null) {
    iconoMenu.setTint(ContextCompat.getColor(this, R.color.white));
  }
}
```

Responder a los eventos del menú

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.op1){
        Toast.makeText(this, "Se seleccionó la primera opción", Toast.LENGTH_LONG).show();
        return true;
    }
    if (id == R.id.op2){
        String mensaje = "Se seleccionó la segunda opción";
        if (item.isChecked()) {    // Si el checkbox ya estaba activado
            item.setChecked(false); // Se desactiva
            mensaje += " y se desactivó";
        } else {
            item.setChecked(true);  // Si no estaba activado se activa
            mensaje += " y se activó";
        }
        Toast.makeText(this, mensaje, Toast.LENGTH_LONG).show();
        return true;
    }
    if (id == R.id.op3){
        Toast.makeText(this, "Se seleccionó la tercera opción", Toast.LENGTH_LONG).show();
        return true;
    }
    if (id == R.id.op4 || id == R.id.op5){
        item.setChecked(true);
        Toast.makeText(this, "Se seleccionó un radiobutton", Toast.LENGTH_LONG).show();
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Acceso a los ítems del menú

Para poder acceder a los ítems del menú desde fuera de `onOptionsItemSelected()` es necesario haber guardado una referencia al crearlos

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.barraherramientas, menu);
    menuOpciones = menu; // menuOpciones es un atributo Menu
    return true;
}
```

Posibles operaciones que se pueden hacer con los ítems del menú:

```
// Obtener el estado de un ítem seleccionable
boolean estado = menuOpciones.findItem(R.id.op2).isChecked();
// Establecer el estado de un ítem seleccionable
menuOpciones. findItem(R.id.op2).setChecked(true);
// Deshabilitar un ítem (no ejecutar el manejador):
menuOpciones.findItem(R.id.op2).setEnabled(false);
// Ocultar todos los items:
for (int i = 0; i < menuOpciones.size(); i++)
    menuOpciones.getItem(i).setVisible(false);
```

Activar regreso a la actividad padre

- En una APP con varias actividades, puede ser deseable ofrecer una forma de regresar a la actividad padre para no usar necesariamente el botón de vuelta atrás del terminal

- Esto se puede conseguir en una *ActionBar* invocando al método `setDisplayHomeAsUpEnabled()`

```
ActionBar actionBar=getSupportActionBar();
actionBar.setDisplayHomeAsUpEnabled(true); // Muestra el icono UP (flecha ←)
```

- Para cambiar el color del icono UP:

```
Drawable flecha = barraHerramientas.getNavigationIcon();
if (flecha != null) {
   flecha.setTint(ContextCompat.getColor(this, R.color.white));
}
```

Gestionar regreso a la actividad anterior

Si se pulsa el icono UP de la *ActionBar*, se puede gestionar su comportamiento con alguna de las siguientes alternativas:

1. Tratándolo como un elemento más de la *ActionBar* en `onOptionsItemSelected()`:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == android.R.id.home){// Se ha pulsado el icono UP (flecha ←)
        Toast.makeText(this, "Se seleccionó botón atrás",
                                    Toast.LENGTH_LONG).show();
        ::::
        finish();
        return true;
    }
    ::::
```

Gestionar regreso a la actividad anterior

2. Sobrescribiendo el método `onSupportNavigateUp()`

```
@Override
public boolean onSupportNavigateUp() {
   // Personalizar el comportamiento, si se desea
   finish();
   return true;
}
```

Gestionar regreso a la actividad anterior

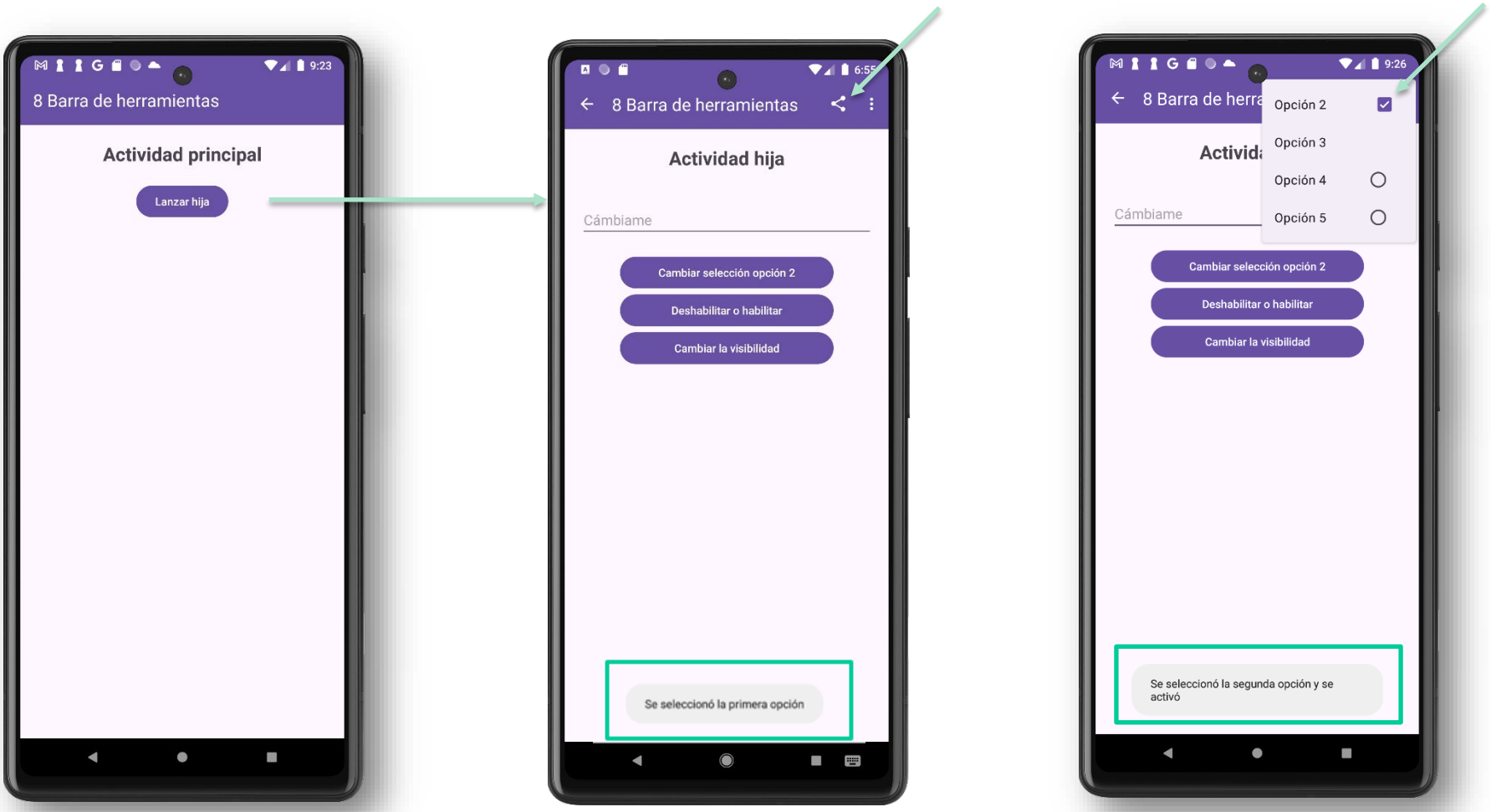
- Si se quiere cambiar el comportamiento del botón físico de vuelta atrás hay que sobrescribir el método `handleOnBackPressed()` de la clase `OnBackPressedCallback`.

```
OnBackPressedCallback manejadorBotonAtras = new OnBackPressedCallback(true) {
    @Override
    public void handleOnBackPressed() {
        // Poner el nuevo comportamiento de la acción de volver atrás. Por ejemplo
        // mostrar un mensaje con un toast y terminar
        Toast.makeText(getApplicationContext(), "Has tocado el botón de vuelta atrás",
                                                Toast.LENGTH_LONG).show();
        setEnabled(false); // Deshabilita este callback (solo si se necesitara)
        finish(); // Finaliza la actividad actual (si es lo que se desea)
    }
};
getOnBackPressedDispatcher().addCallback(this, manejadorBotonAtras);
```

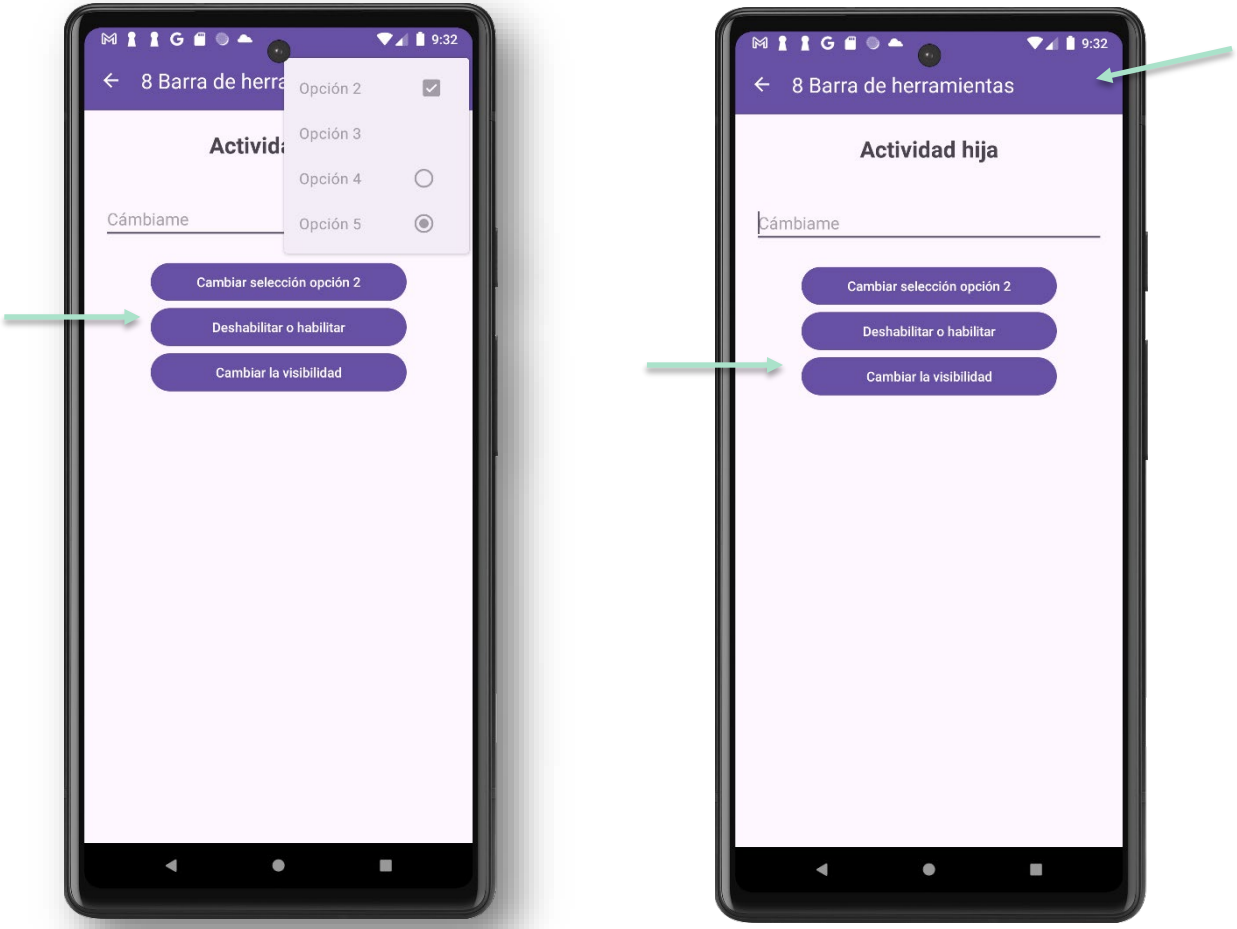
Ejercicio 1 (1)

- Crea un proyecto nuevo, con las siguientes características:
  - Nombre: “8 Barra de herramientas”
  - Paquete: dam.barraherramientas
  - Directorio: “8-BarraHerramientas”
- Crea una aplicación con dos actividades. Ambas deben usar un tema sin *ActionBar*
- Desde la actividad inicial se lanzará la segunda actividad al pulsar un botón
- La segunda actividad debe tener una *Toolbar*, como la que se muestra en la figura, con este comportamiento:
  - Debe tener el icono para regresar a la actividad principal
  - Debe tener una opción con icono en la barra de herramientas
  - Debe tener cuatro opciones en el menú, una seleccionable y dos de tipo *radiobutton*
  - Debe tener 3 botones para realizar la acción que se indica en los nombres de estos
  - Al pulsar las opciones, se mostrará con un *Toast* la opción seleccionada
  - Debe tener un *EditText* cuyo valor se recoja al pulsar el botón de vuelta atrás
  - Al pulsar el botón físico no se debe volver a la actividad anterior, indicándolo con un *Toast*
- Al volver a la actividad inicial se mostrará el texto escrito en el *EditText* de la segunda actividad

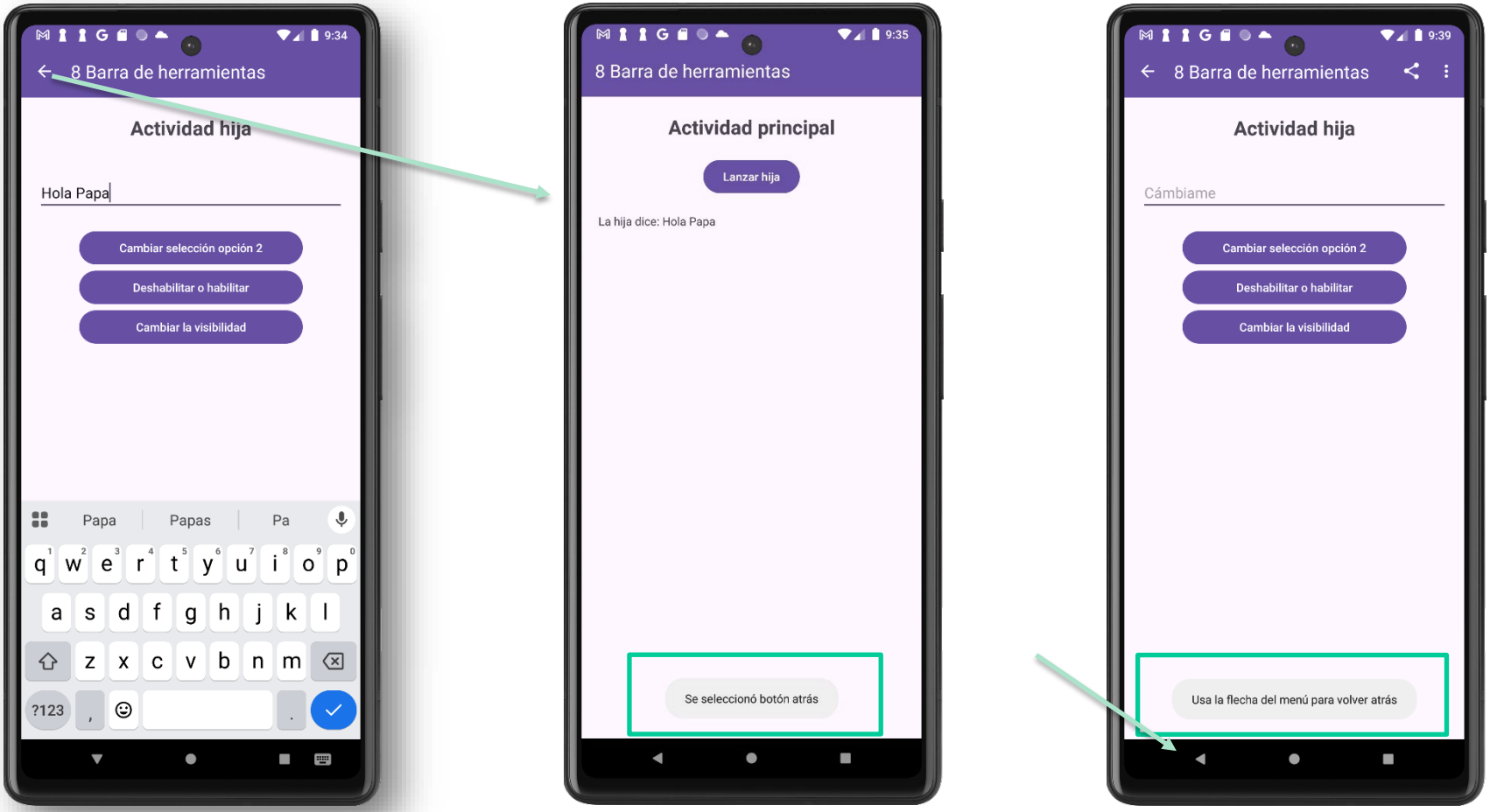
Ejercicio 1 (2)



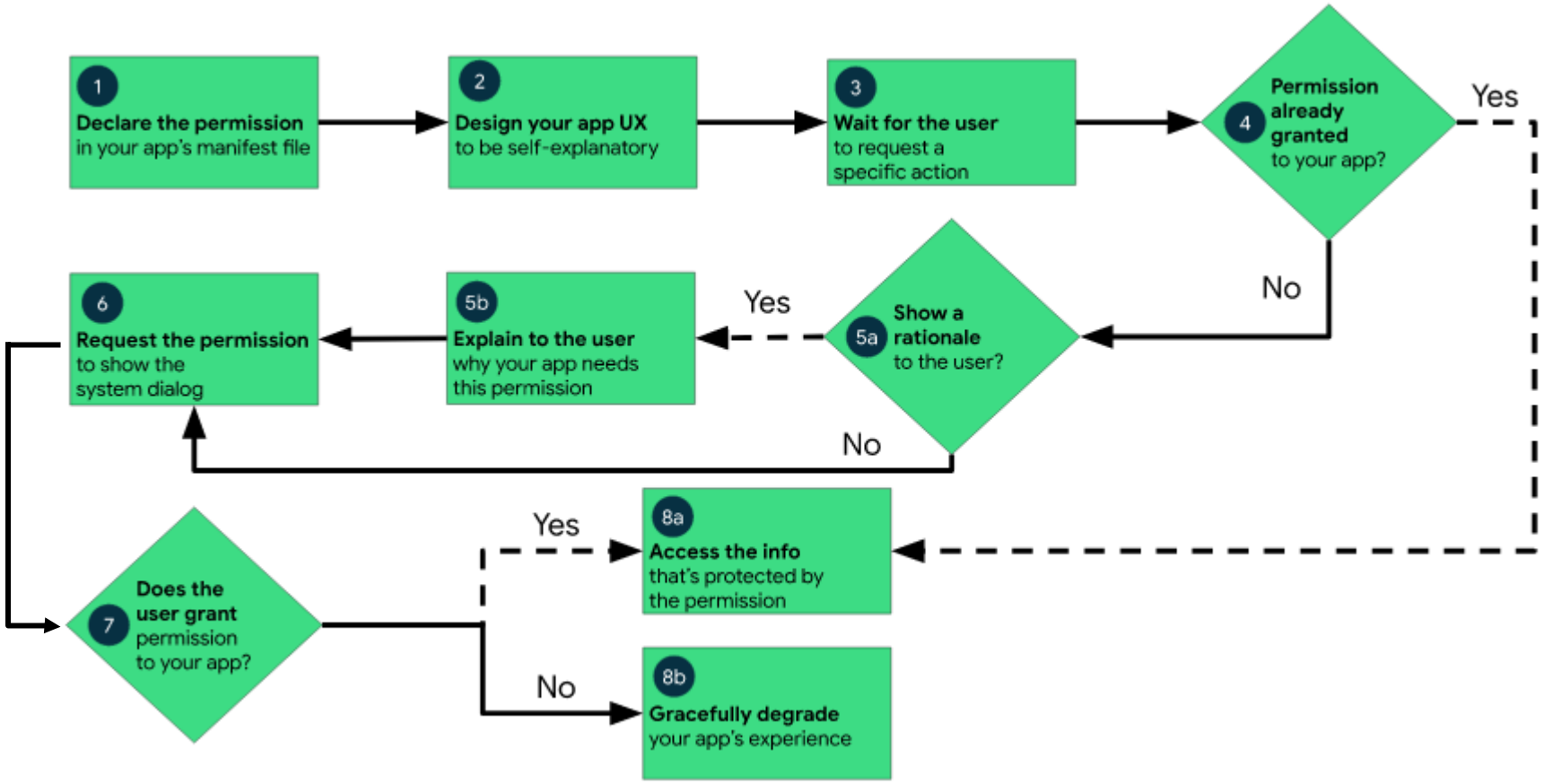
Ejercicio 1 (3)



Ejercicio 1 (4)



Permisos: flujo para solicitarlos



https://developer.android.com/training/permissions/requesting?#workflow_for_requesting_permissions

Permisos: declaración en el manifiesto

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="paquete">

    <uses-permission android:name="android.permission.PERMISO_NECESARIO" />
    <uses-permission android:name="android.permission.OTRO_PERMISO" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

https://developer.android.com/training/permissions/declaring#add-to-manifest

Permisos: petición y resultado

- `checkSelfPermission()`: verifica si la actividad tiene concedido un permiso
- `shouldShowRequestPermissionRationale()`: devuelve *true* si el usuario denegó la solicitud de un permiso en una petición anterior
- `ActivityResultLauncher.launch()`: solicita permiso
- `ActivityResultContracts.RequestPermission()`: invocado cuando el usuario responde a la petición de permisos realizada en `ActivityResultLauncher.launch()`

Permisos. Ejemplo de código (1)

```
// Definición de un atributo con la referencia del código a ejecutar cuando se solicita el permiso
private ActivityResultLauncher<String> lanzadorPeticionPermiso =
    registerForActivityResult(
        new ActivityResultContracts.RequestPermission(),   // Clase que muestra el cuadro de diálogo para pedir permiso
        esConcendido -> {            // Función a ejecutar cuando se retorna del cuadro de diálogo devolviendo el resultado
            if (esConcendido) {
                // Permiso concedido, se puede ejecutar el código que lo necesita
                operacionConPermisos();
            } else {
                // Permiso denegado, indicarlo al usuario
            }
        });
```

https://developer.android.com/training/permissions/requesting?#allow-system-manage-request-code

Permisos. Ejemplo de código (2)

```
private void verificarOsolicitarPermisos(){

    // Verificar si ya se tiene concedido el permiso de otra petición anterior
    if (ContextCompat.checkSelfPermission(this,
        android.Manifest.permission.PERMISO_NECESARIO) == PackageManager.PERMISSION_GRANTED) {
          // Si tiene el permiso concedido, invocar al método que realiza la operación
          operacionConPermisos();

    } else {

          // No tiene el permiso, entonces solicitarlo

          // Antes de solicitarlo, verificar si ya se solicitó anteriormente
          if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                                                    android.Manifest.permission.PERMISO_NECESARIO)){
              solicitarPermisoDeNuevo();

          } else {
              // Si ha sido la primera vez que se va a solicitar el permiso, hacer la solicitud
              solicitarPermiso(this);
          }
}
```

Permisos. Ejemplo de código (3)

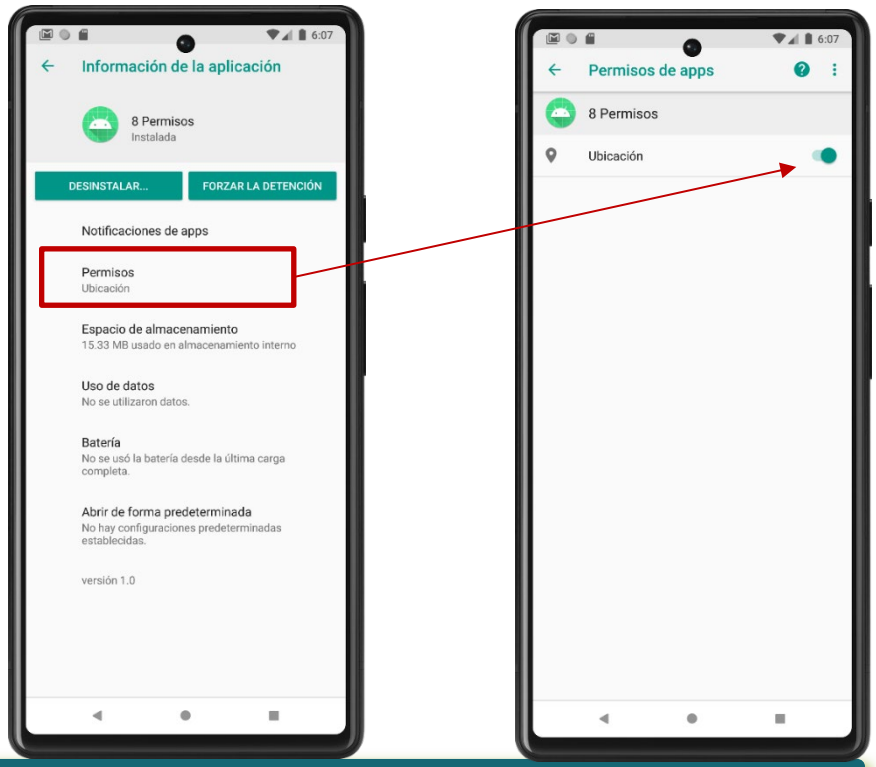
```
private void solicitarPermiso(Activity actividad) {
        lanzadorPeticionPermiso.launch(android.Manifest.permission.PERMISO_NECESARIO);
}

private void solicitarPermisoDeNuevo(){
        final Activity actividad=this;
        // Mostrar un cuadro de diálogo explicando el motivo de solicitar el permiso
        new AlertDialog.Builder(actividad)
                .setTitle("Solicitud de permiso")
                .setMessage("Justificación: este permiso se le solicita por ....")
                . setNegativeButton("No acepto", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                                // El usuario sigue sin aceptar  el permiso
                        }
                })
                .setPositiveButton("Entendido", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int whichButton) {
                                solicitarPermiso(actividad)
                        }
                })
                .show(); // Mostrar el cuadro de diálogo
}
```

https://developer.android.com/reference/androidx/core/app/ActivityCompat#requestPermissions(android.app.Activity,java.lang.String[],int)

Retirar un permiso a una aplicación

- En cualquier momento (incluso con la aplicación en ejecución) el usuario puede retirar un permiso que anteriormente fuera concedido
- Ejemplo de cómo retirar el permiso de ubicación a una aplicación llamada *Cámaras Madrid*:

En los ajustes de Android, opción *Aplicaciones y notificaciones*, seleccionar la aplicación, y en la ventana de características, hacer clic en *Permisos* y en la siguiente pantalla deshabilitarlo

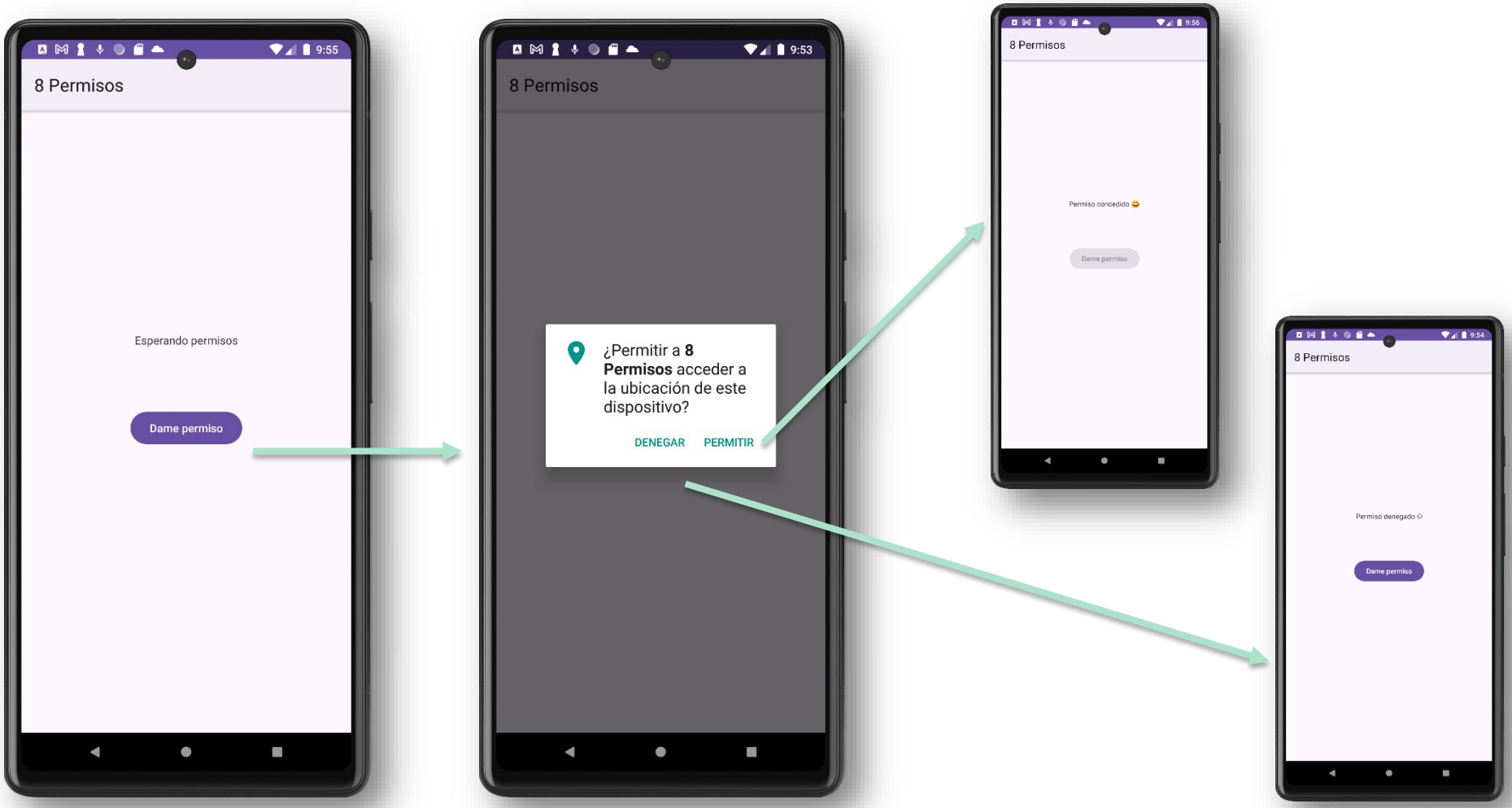


Ejercicio 2 (1)

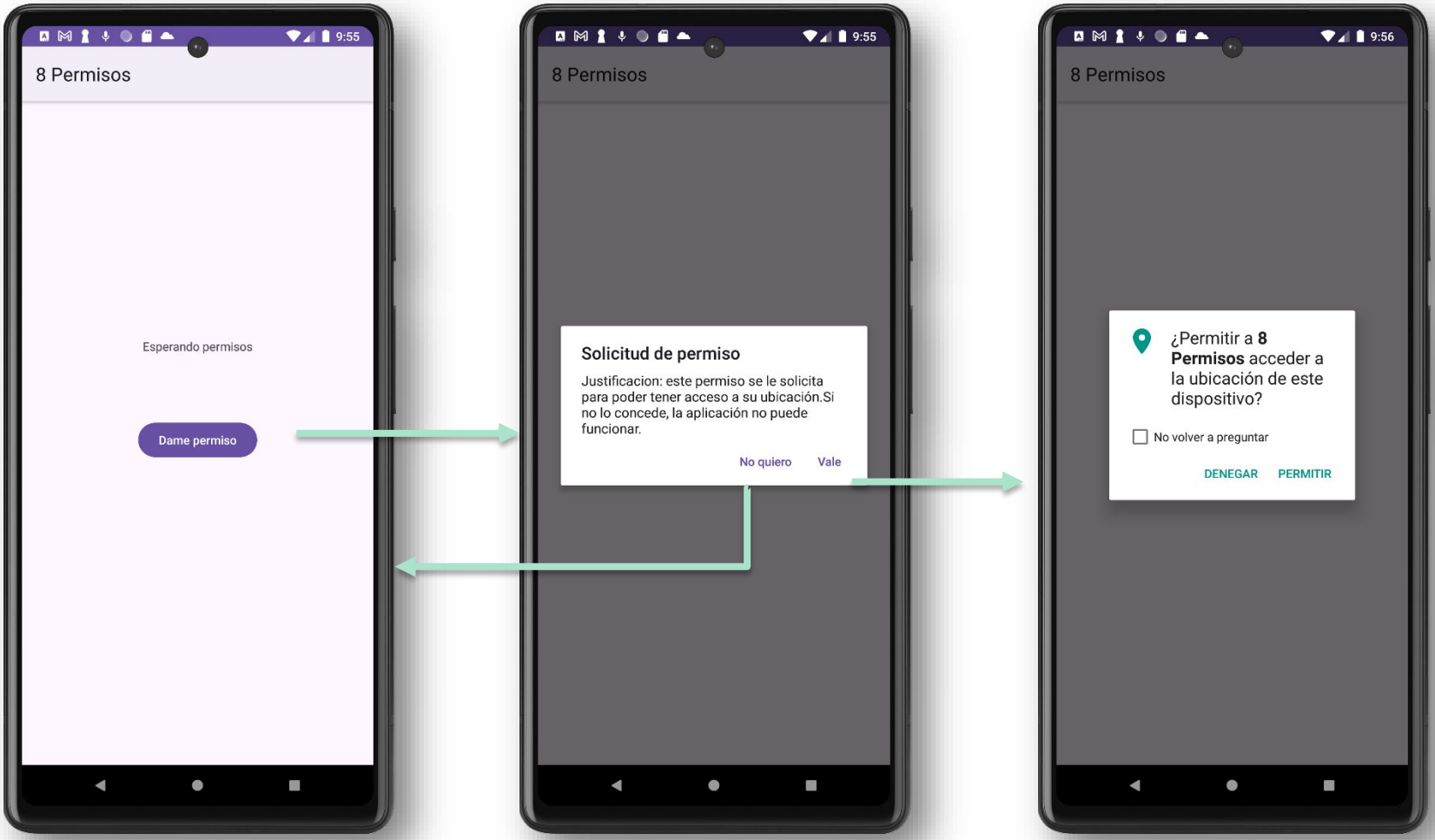
- Crea un proyecto nuevo, con las siguientes características:
  - Nombre: “8 Permisos”
  - Paquete: dam.permisos
  - Directorio: “8-Permisos”
- La actividad muestra lo siguiente:
  - Un `TextView` que indica si tiene o no el permiso de localización
  - Un botón que al hacer clic solicita el permiso de localización
- Tiene este comportamiento:
  - Si es la primera vez que se va a solicitar el permiso, al pulsar el botón se le solicita sin más, indicando en el `TextView` si se le ha concedido
  - Si ya se había solicitado previamente el permiso, al pulsar el botón se informa al usuario del motivo del permiso y se le vuelve a pedir el permiso
  - Al arrancar verifica si tiene concedido el permiso de localización*. Si lo tiene lo indica y muestra el botón deshabilitado

* El permiso de localización es ACCESS_FINE_LOCATION, si bien en el *manifest* también hay que añadir ACCESS_COARSE_LOCATION, aunque solo se debe pedir el primero

Ejercicio 2 (2)



Ejercicio 2 (3)



Ejercicio 2 (4)

Arrancar la aplicación cuando ya tiene el permiso concedido de otra ejecución

Servicio de localización

- Disponible a través de los servicios de Google Play
- Sólo se puede obtener la última ubicación conocida, aunque se puede solicitar actualizar la ubicación
- Existen varios proveedores de localización:
  - Por GPS
  - Por la red
- Puede usarse un proveedor combinado para no especificar un proveedor concreto
- La APP necesita el permiso ACCESS_COARSE_LOCATION (aproximada) o ACCESS_FINE_LOCATION (precisa)

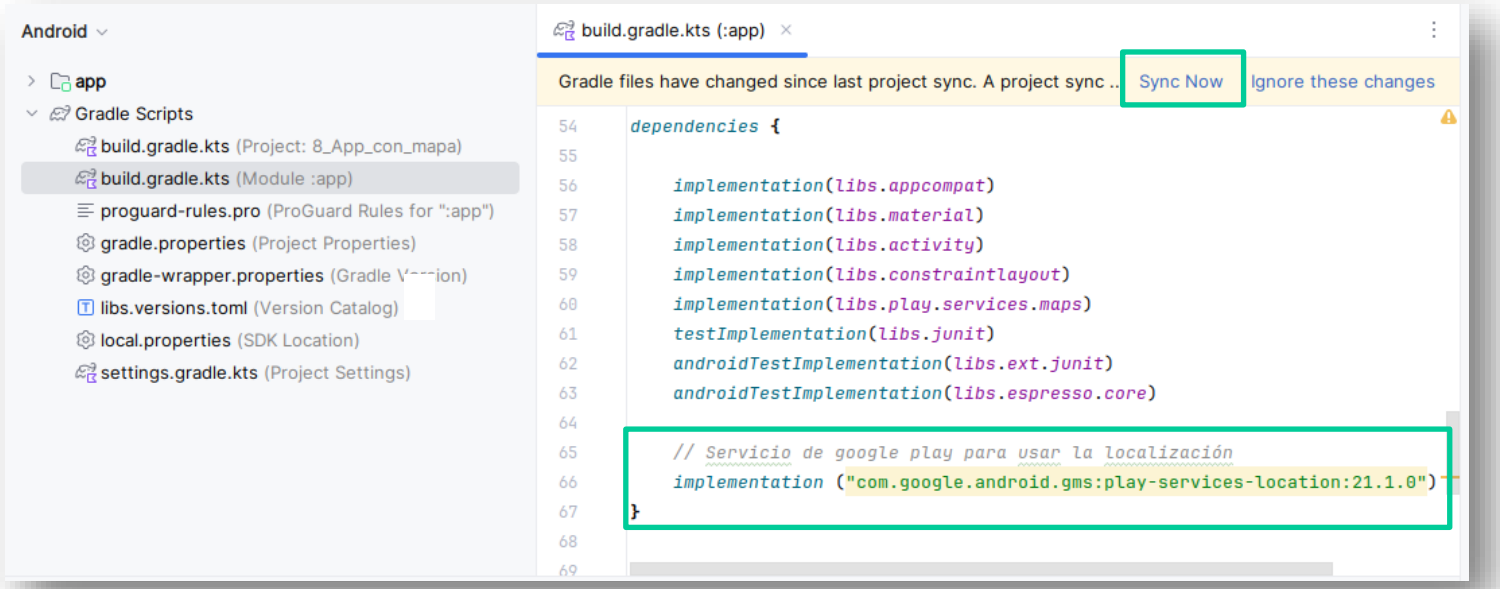
https://developer.android.com/training/location

Obtener la localización

```
FusedLocationProviderClient clienteLocalizacion;
clienteLocalizacion = LocationServices.getFusedLocationProviderClient(this);

// Obtener de forma asíncrona la localización mediante getLastLocation()
// También se puede usar getCurrentLocation() que obtiene una ubicación mas actualizada, pero
// a costa de gastar más batería
Task<Location> ultimaLocalizacion = clienteLocalizacion.getLastLocation();

ultimaLocalizacion.addOnSuccessListener(this, new OnSuccessListener<Location>() {
            @Override
            public void onSuccess(Location localizacion) {
                // Se ha obtenido la localización
                // Verificar que no es nula, esto podría suceder la primera vez si nunca se ha usado
                // el servicio de localización en el terminal o está desactivada en el terminal

                if (localizacion != null) {
                      // Localización encontrada
                      // Obtener latitud: localizacion.getLatitude()
                      // Obtener longitud: localizacion.getLongitude()
                } else {
                       // Localización no encontrada
                }
            }
        });
```

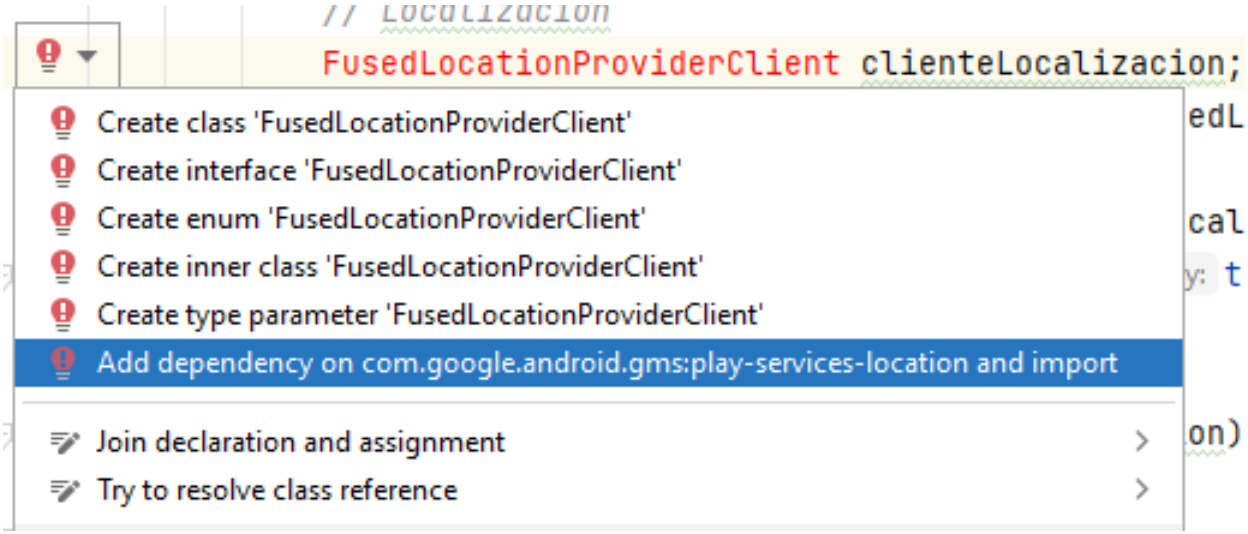
Declarar la dependencia del servicio de localización de Google Play (2)

Se necesita declarar la dependencia del servicio `com.google.android.gms:play-services-location` en el fichero build.gradle



Declarar la dependencia del servicio de localización de Google Play (2)

Una forma cómoda de hacerlo es con el asistente:



https://developers.google.com/android/guides/setup#dependencies

https://developers.google.com/android/guides/releases?authuser=3

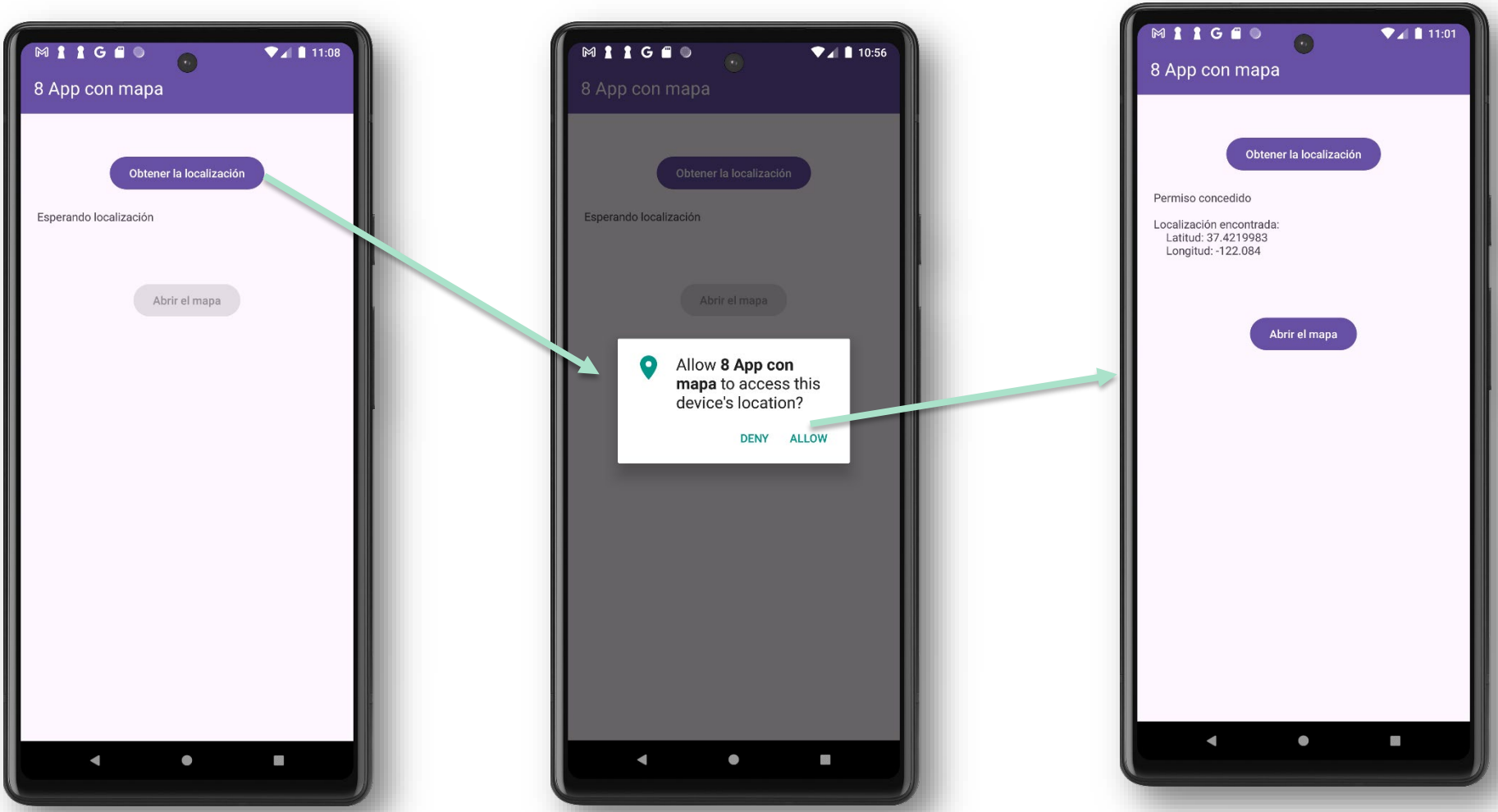
Ejercicio 3 (1)

- Crear un proyecto nuevo, con las siguientes características:
  - Nombre: “8 App con mapa”
  - Paquete: dam.appmapa
  - Directorio: “8-AplicacionMapa”
- Crear una aplicación con dos actividades. Desde la actividad inicial se lanzará la segunda actividad al pulsar un botón
- La actividad inicial:
  - Al iniciarse debe obtener la localización actual y mostrarla. Si no tiene el permiso de localización debe solicitarlo y mostrar el resultado de la petición
  - Debe mostrar un botón para lanzar la segunda actividad que solo estará habilitado si ha podido obtener la localización
- La segunda actividad
  - Debe mostrar un mapa con un marcador en la localización obtenida
  - En la barra de título debe mostrar el botón para regresar a la actividad principal*

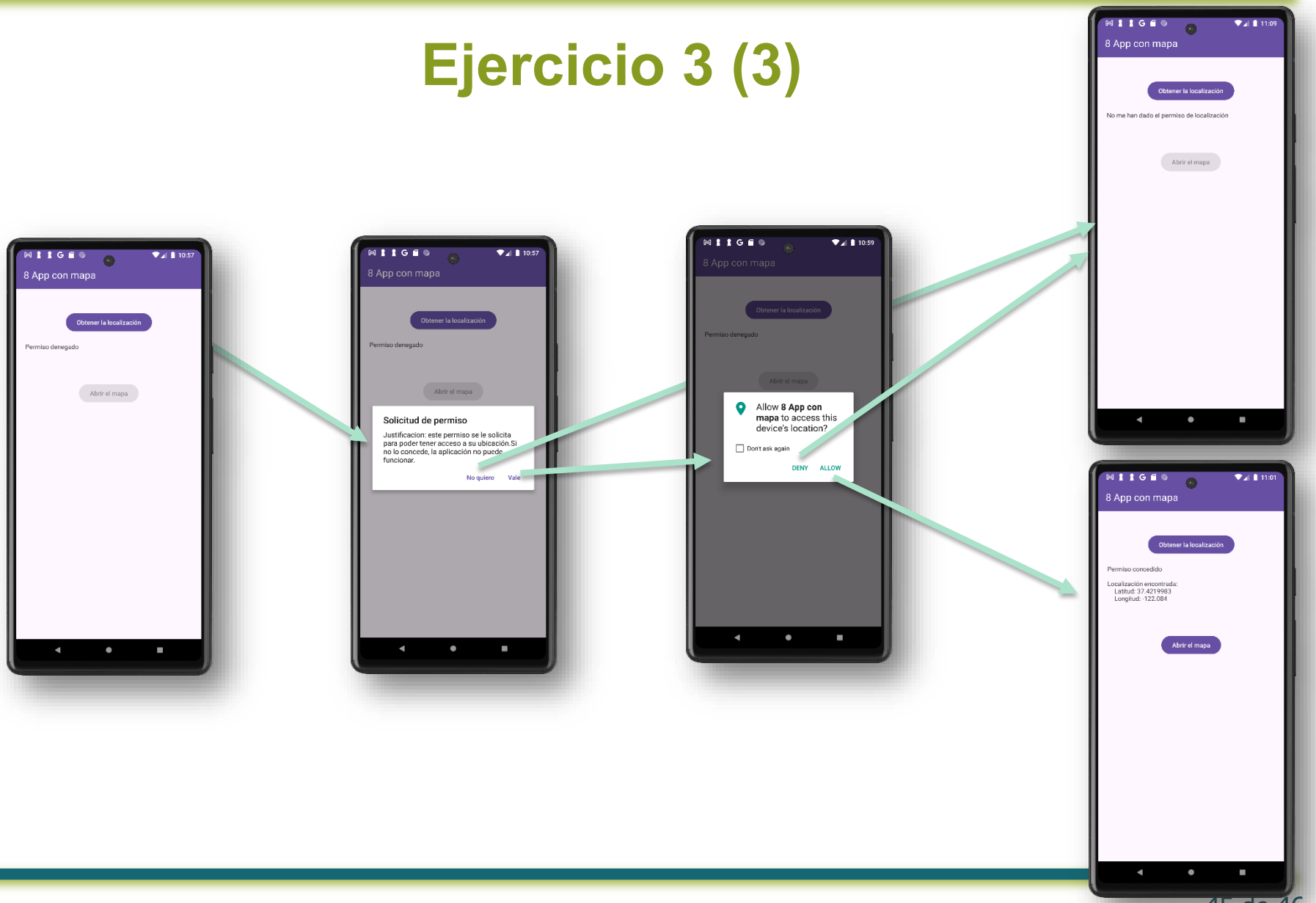
* Para poder usar una *Toolbar* en la actividad del mapa, esta debe extender de `AppCompatActivity` en vez de `FragmentActivity`.

Además, como la actividad no tiene menus, usar el método onSupportNavigateUp() para implementar la vuelta atrás

Ejercicio 3 (2)



Ejercicio 3 (3)



Ejercicio 3 (4)