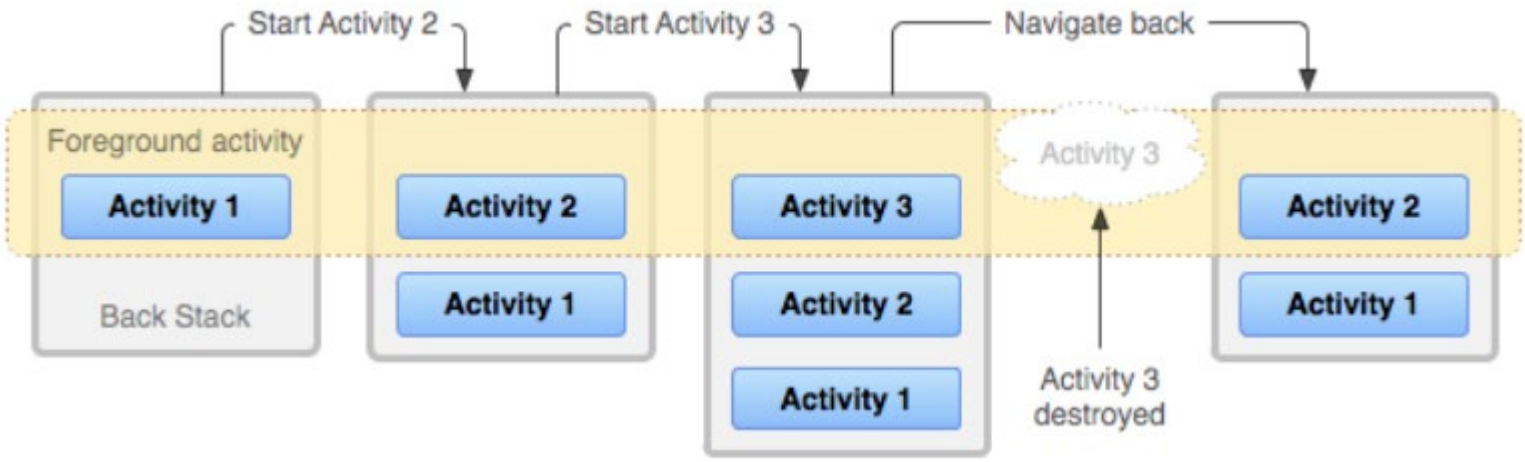
SESIÓN 5

Múltiples actividades

Persistencia

Actividades

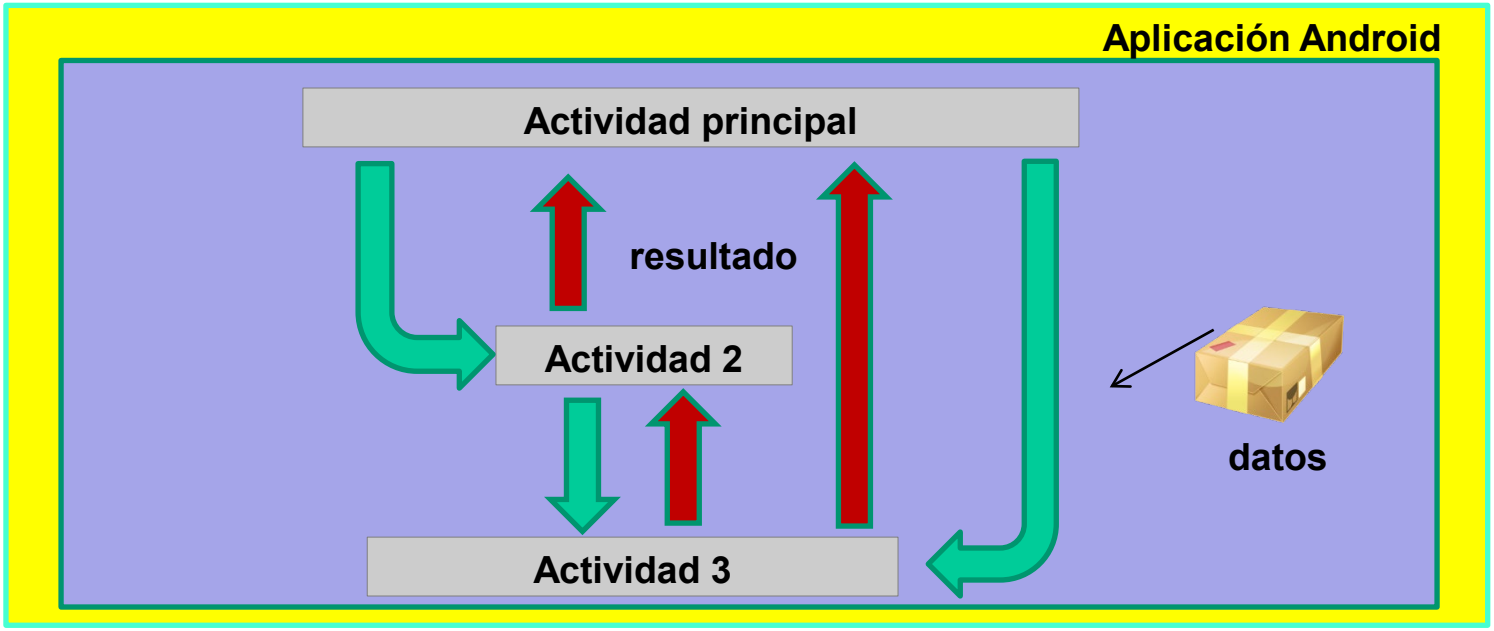
- Una aplicación está compuesta por un conjunto de actividades independientes. Cada actividad representa la unidad de interacción con el usuario (una pantalla)
- La navegación por las actividades de la aplicación se organiza en forma de pila. Normalmente se retorna a la actividad llamante gracias a un botón (físico, en pantalla o un gesto).
- Cuando una actividad inicia otra actividad, esta pasa a primer plano (visible) y la otra a segundo plano. Cuando se pulsa el botón atrás, la nueva actividad se destruye (sale de la pila) y se muestra la primera actividad que vuelve a primer plano.



https://developer.android.com/guide/components/activities/tasks-and-back-stack

Intents (1)

- Un *intent* es una descripción abstracta de una operación a realizar
- En un *intent* se puede especificar la actividad que realizará la operación y datos que necesite
- También se usa para dejar los resultados que devuelve la actividad invocada.



https://developer.android.com/guide/components/intents-filters

Intents (2)

- Se pueden lanzar una nueva actividad mediante **`startActivity()`** o **`ActivityResultLauncher.launch()`** (si se espera recibir algún resultado). Entre los argumentos de estos métodos está el *intent* que define qué actividad se va a lanzar.
- Los *intents* pueden ser implícitos y explícitos:
  - Los **implícitos**, piden al S.O. llevar a cabo una acción genérica, por ejemplo, abrir un navegador, hacer una foto, hacer una llamada telefónica … Por tanto, no especifican la actividad a lanzar
  - Los **explícitos**, sirven para iniciar una actividad específica, normalmente de la aplicación actual.

Intents implícitos (1)

1. Crear un *intent* especificando la acción a llevar a cabo, ejemplo mostrar datos:

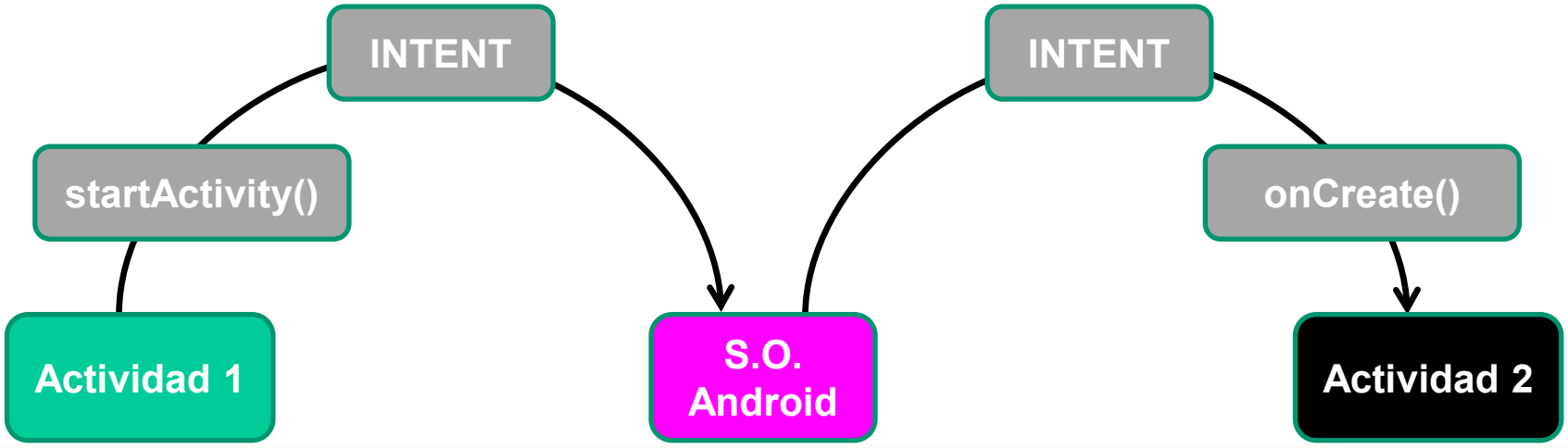
```
Intent intent = new Intent(); intent.setAction(Intent.ACTION_VIEW);
```

2. Añadir al *intent* algún dato, por ejemplo, el url de la página a mostrar:

```
intent.setData(Uri.parse("https://www.google.com"));
```

3. Iniciar una actividad preparada para esa acción: `startActivity(intent);`

Puede haber varias actividades que lleven a cabo la acción, por ejemplo, si hay varios navegadores instalados, se invocará el de por defecto o se le pedirá al usuario que elija uno.



Intents implícitos (2)

- Un *intent* implícito está formado por:
  - Acción genérica que llevar a cabo, que identificará posibles actividades a lanzar
  - Datos con los que operar, expresados como una URI
  - Otra información: categoría, tipo, extras, …
- Ejemplo de acciones (constantes definidas en la clase `Intent`):

Acción	URI	Descripción
Intent.ACTION_VIEW	https://www.google.com	Lanza un navegador en el url especificado
Intent.ACTION_VIEW	geo:40.3895417, -3.6285383	Lanza una aplicación de mapa en una posición determinada
Intent.ACTION_CALL	tel:555-555-555	Lanza la aplicación de teléfono para que haga una llamada al nº especificado

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("geo:40.3895417,-3.6285383"));
startActivity(intent);
```

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_VIEW)
```

- Lista completa de acciones:

http://developer.android.com/reference/android/content/Intent.html #constants_1

Intents implícitos (3)

- Un *intent* puede llevar *Extras*, que son argumentos formados por parejas clave/valor. Se usa el método `putExtra(clave,valor)`
- También se puede indicar el tipo de datos que se envían/reciben. Se usa el método `setType(`*tipo MIME*`)`
- Ejemplo para enviar un correo:

```
final Intent intent = new Intent(Intent.ACTION_SENDTO);
intent.setData(Uri.parse("mailto:"));
intent.putExtra(Intent.EXTRA_EMAIL,
        new String [] {"un@destinatario.es", "dos@destinatario.es"});
intent.putExtra(Intent.EXTRA_SUBJECT, "Este es el asunto");
intent.putExtra(Intent.EXTRA_TEXT, "Este es el contenido");
intent.setType("text/plain");
startActivity(intent);
```

https://developer.android.com/training/basics/intents/sending

Intents implícitos (4)

- De forma genérica, la actividad que quiera lanzar de forma implícita otra actividad usará este código (la palabra "ACCION" es una constante definida en la clase *Intent*):

```
final Intent intent = new Intent("Intent.ACCION");
intent.setType("tipo");
intent.setData(datos);
intent.putExtra(Intent.CLAVE, "valor");
// Verificar que el intent puede ser resuelto por alguna actividad
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

- O bien este, donde se fuerza a usar el selector de apps:

```
final Intent intent = new Intent("Intent.ACCION");
...
final Intent selector = Intent.createChooser(intent, "Texto ayuda");
// Mostrar el selector de apps para que el usuario elija
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(selector);
}
```

Ejercicio 1

- Crea un proyecto nuevo, con las siguientes características:
  - Nombre: “5 Intents implícitos”
  - Paquete: dam.intents
  - Directorio: 5-IntentsImplicitos
- La aplicación debe disponer de tres botones para lanzar las siguientes aplicaciones:
  - Aplicación de mapa en la localización de tu domicilio, con zoom de 16, marcador y etiqueta. Ayuda
  - Navegador en el url especificado en el `EditText`. Debe mostrar el selector de apps para que el usuario elija navegador
  - Cliente de correo con los datos que se le proporcionen en los `EditText`

Ver ejemplos aquí:

https://developer.android.com/guide/components/intents-common

Intents implícitos (5)

- Para que una actividad pueda ser lanzada usando un *intent* implícito, debe registrar, en el fichero `AndroidManifest.xml`, uno o más **filtros** con la acción y categoría que permite invocarla, y opcionalmente datos:

```
<intent-filter>
    <action android:name="android.intent.action.ACCION"/>
    <category android:name="android.intent.category.CATEGORIA"/>
    <data android:tipo1="tipo1" android:tipo2="tipo2" …/>
</intent-filter>
```

- Las palabras "`ACCION`" y "`CATEGORIA`" son constantes de la clase *Intent*, donde se definen las acciones y categorías genéricas
  https://developer.android.com/reference/android/content/Intent#constants_1

- El elemento `data` se usa para determinar la aplicación más adecuada
  https://developer.android.com/guide/topics/manifest/data-element

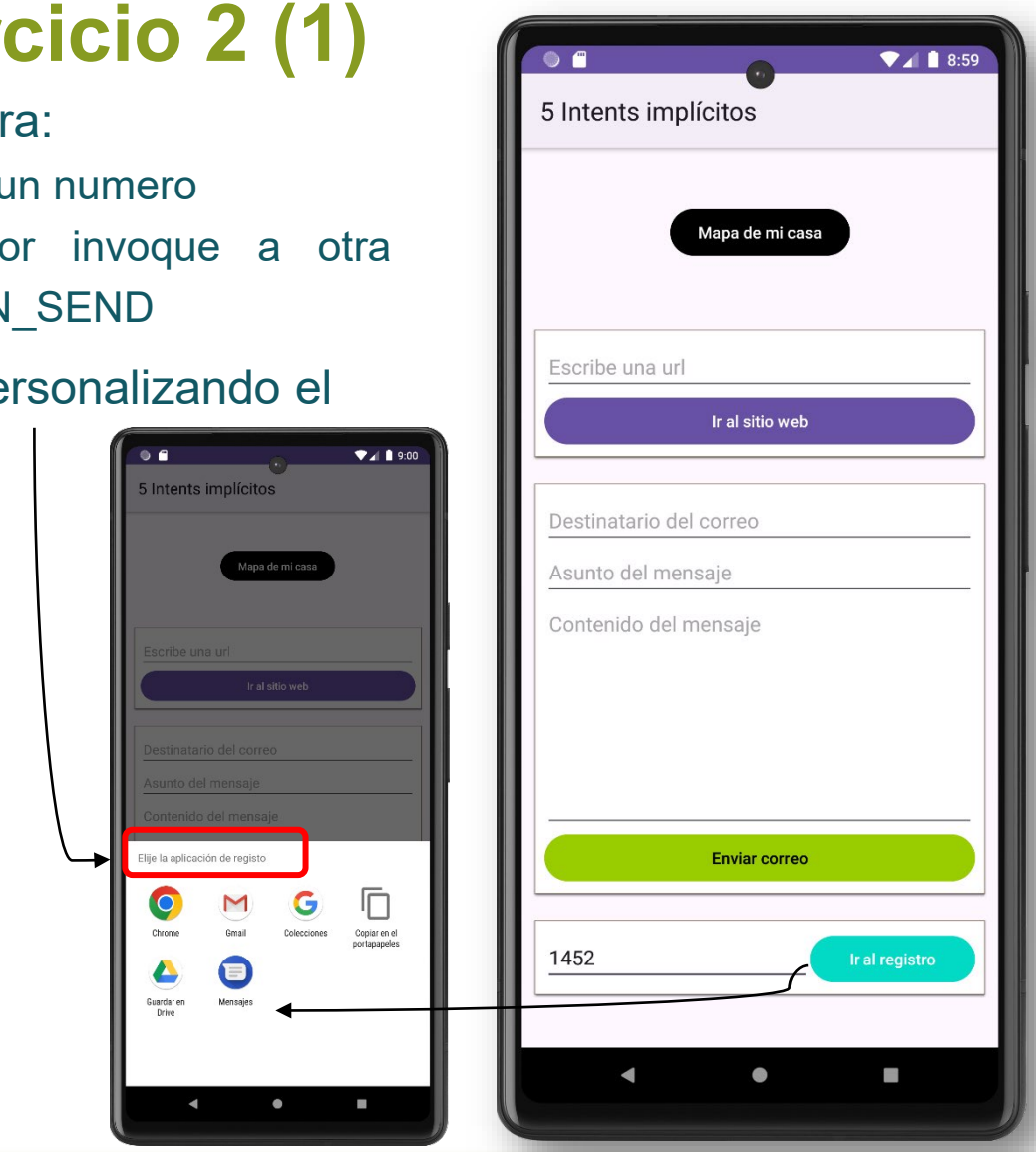
Intents implícitos (6)

Ejemplo del fichero **`AndroidManifest.xml`** de una aplicación que tiene una actividad que admite un *intent* implícito genérico:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <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" android:exported="true" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
            <intent-filter>
              <!- Filtro añadido para admitir un intent implícito para compartir texto -->
              <action android:name=" android.intent.action.SEND"/>
              <category android:name="android.intent.category.DEFAULT"/>
              <data android:mimeType="text/plain"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Ejercicio 2 (1)

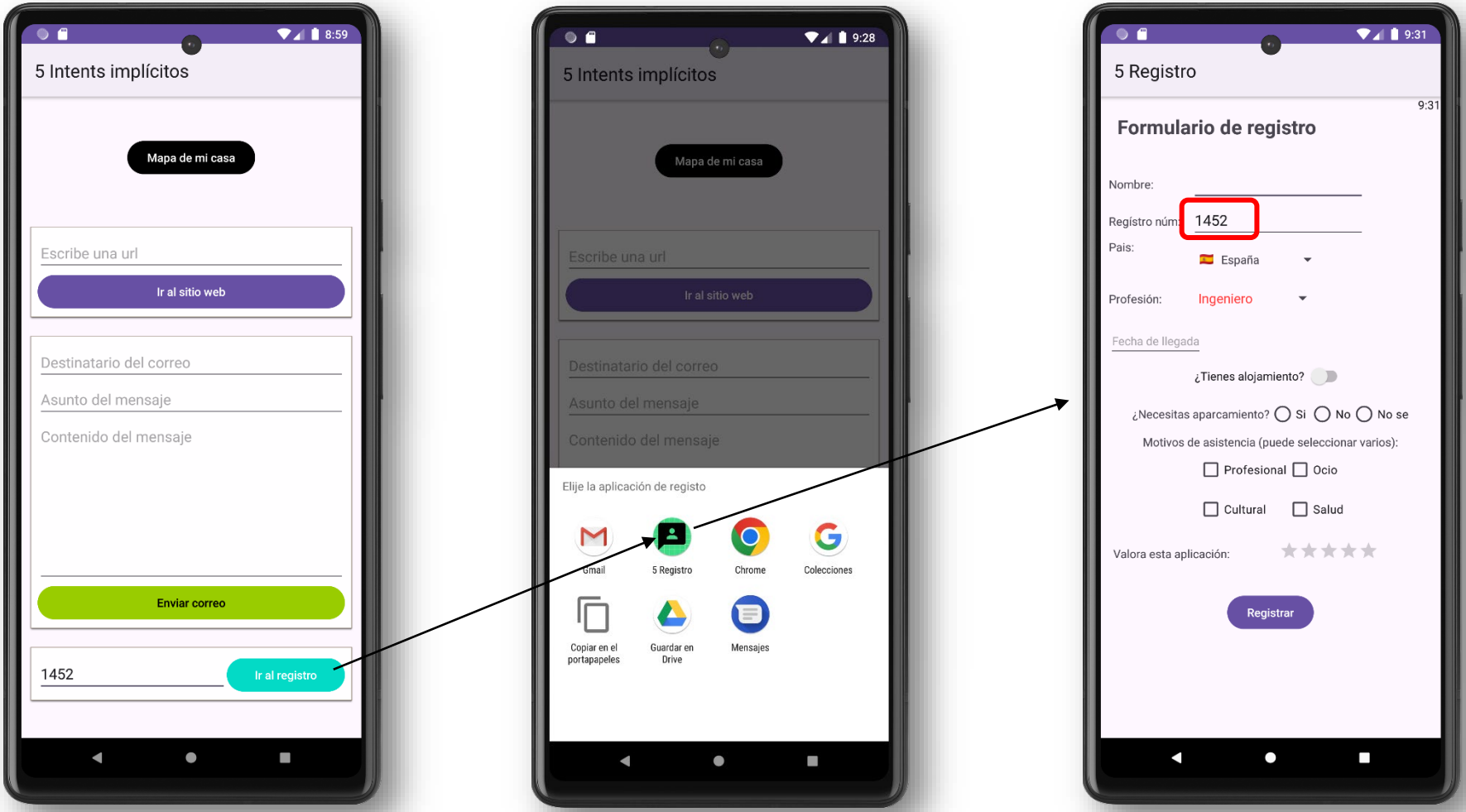
- Modificar la aplicación anterior para:
  - Añadir un `EditText` donde escribir un numero
  - Añadir un botón cuyo manejador invoque a otra aplicación usando la acción ACTION_SEND
- Puedes usar este ejemplo, pero personalizando el texto que se muestra en el selector para elegir la aplicación
- Prueba la aplicación y, cuando Android muestre el selector de aplicaciones, elije por ejemplo Gmail



Ejercicio 2 (2)

- A partir del proyecto “4 Registro” de la sesión anterior, crea un proyecto, con las siguientes características:
  - Nombre: “5 Registro”
  - Paquete: dam.registroV3
  - Directorio: Registro
- Modifica su fichero *Manifest* para que pueda recibir datos de texto de otra aplicación. Aquí tienes una ayuda
- Modifica la actividad principal para que, si recibe un texto al arrancar, lo muestre en el `EditText` del campo registro. Aquí tienes un ejemplo
- Observa que si ya tenías una instancia abierta de la app *Registro* (prueba a abrirla e introduce datos) y cambias a la app *Intents implícitos* para invocar a la app *Registro*, se crea una instancia nueva de la app *Registro*

Ejercicio 2 (3)

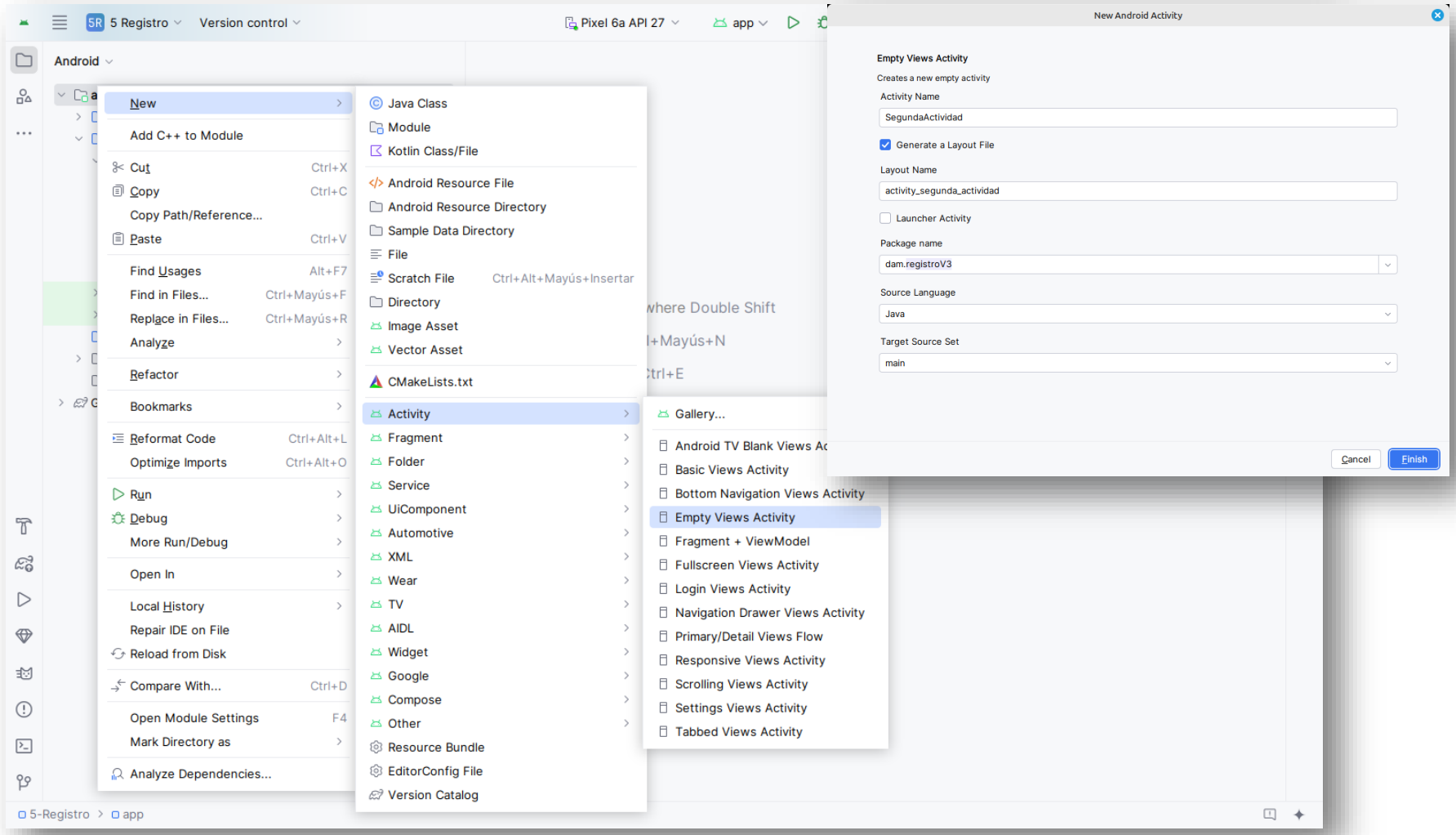


Aplicación con varias actividades

- Cada actividad representa una pantalla de la aplicación y tiene su propio interfaz de usuario (su fichero XML de diseño)
- Las actividades se implementan en clases independientes
- Usan los *intent* como mecanismo de comunicación*:
  - Pueden enviar datos a la actividad que invocan
  - Pueden recibir datos de la actividad que invocan

* Existen otras posibilidades

Añadir una nueva actividad



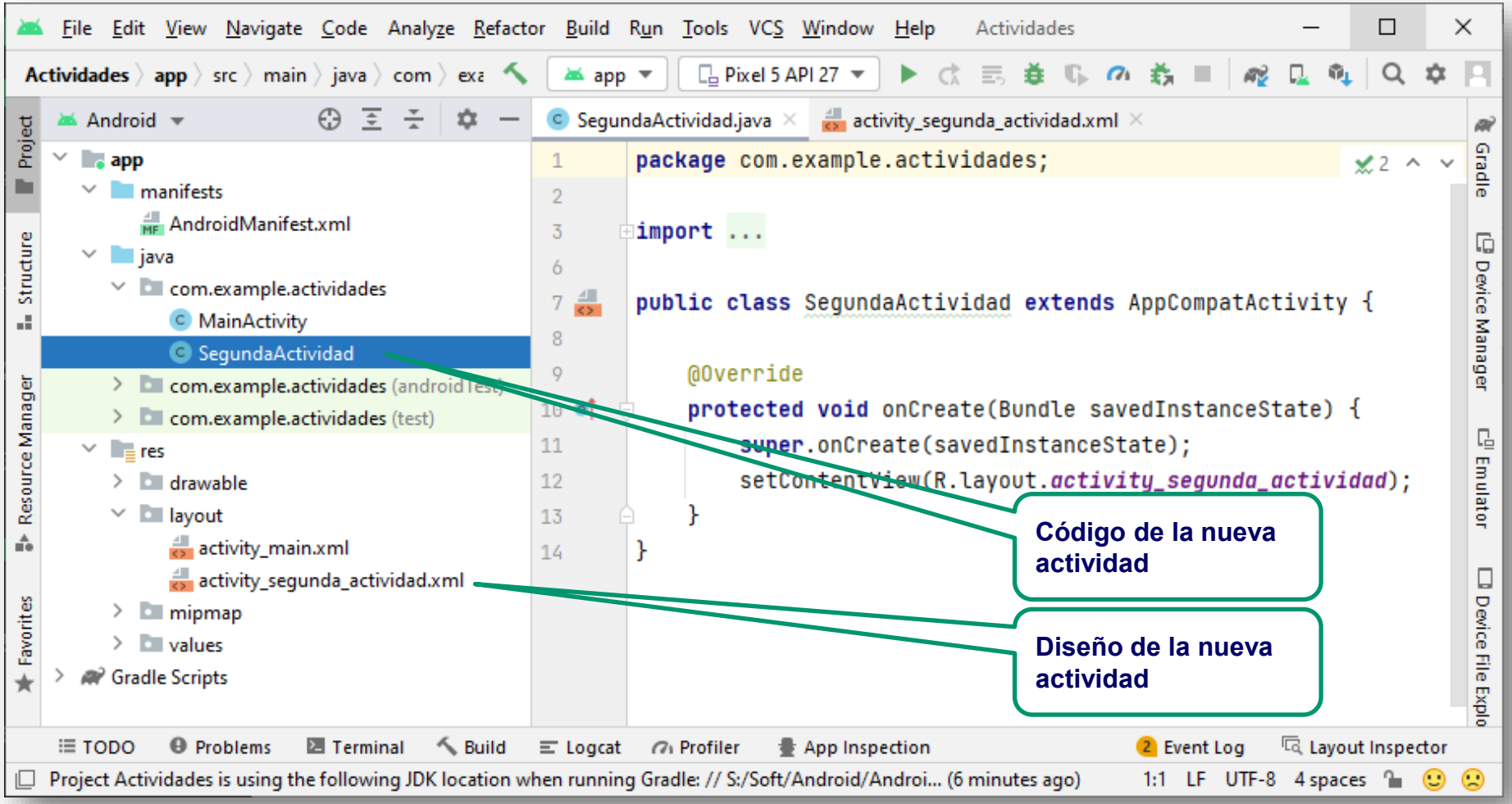
Manifest

Al crear una nueva actividad esta es añadida al fichero **AndroidManifest.xml**

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

    <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/Theme.Actividad">
        <activity android:name=".MainActivity"
                  android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SegundaActividad"
                  android:exported="false" />
    </application>
</manifest>
```

Código y diseño de la nueva actividad



Lanzar una segunda actividad que no devuelve datos

Actividad llamante:

```
final Intent intent = new Intent(this, SegundaActividad.class);
::::
String nombre = nombre_view.getText().toString();
String edad = edad_view.getText().toString();
intent.putExtra("nombre_clave", nombre);
intent.putExtra("edad_clave", Integer.parseInt(edad));
::::::
startActivity(intent);
```

Intent explícito: Actividades llamante y llamada

Paso de parámetros:
- **clave**
- **valor**

Actividad llamada

```
final Bundle parametros = getIntent().getExtras();
String nombre = parametros.getString("nombre_clave");
int edad = parametros.getInt("edad_clave");
```

Recogida de parámetros

Lanzar una segunda actividad que sí devuelve datos (1)

- Actividad llamante:

```
// Definición de un atributo al que se le da un valor
private final ActivityResultLauncher<Intent> resultadosLanzador = registerForActivityResult(
      new ActivityResultContracts.StartActivityForResult(),
      new ActivityResultCallback<ActivityResult>() {
          @Override
          public void onActivityResult(ActivityResult resultado) {
              if (resultado.getResultCode() == Activity.RESULT_OK) {
                 String parametro1 = resultado.getData().getStringExtra("parametro1_clave");
              }
      });

// Cuando se necesite crear la segunda actividad
final Intent intent = new Intent(this, SegundaActividad.class);
::::
resultadosLanzador.launch(intent);
::::
```

Método que se ejecutará cuando termine la segunda actividad para recoger los datos

Lanza la segunda actividad

https://developer.android.com/training/basics/intents/result

Lanzar una segunda actividad que sí devuelve datos (2)

- Actividad llamada

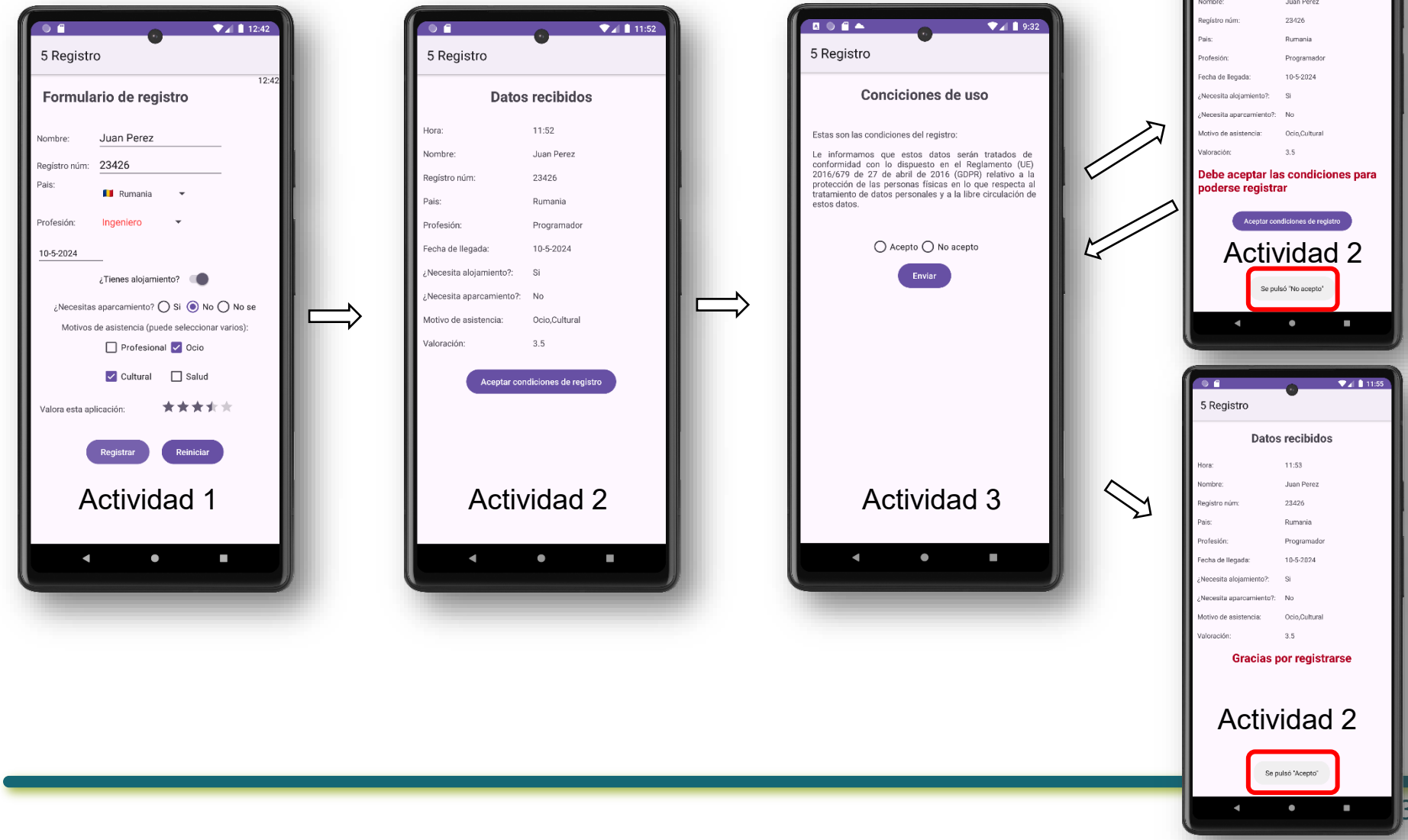
```
Intent intent = new Intent();
intent.putExtra("parametro1_clave", "valor del parámetro 1");
setResult (RESULT_OK, intent);
// setResult (RESULT_CANCELED, intent);
finish();
```

Paso de parámetros a la actividad llamante

Ejercicio 3 (1)

- Modifica la aplicación *Registro* del ejercicio 2, para que muestre **en otra actividad** los datos recogidos por el formulario
- La nueva actividad tendrá un botón para aceptar las condiciones de registro. Al pulsarlo se lanzará **otra actividad** que tendrá dos *RadioButton* (para permitir aceptar o no las condiciones) y un botón para enviar esta información. Cuando se pulse este botón se retornará a la actividad anterior y, en función de la aceptación o no, mostrará un mensaje en un `EditText`:
  - Si acepta se mostrará un mensaje de agradecimiento
  - Si no acepta se mostrará un mensaje que indique que es obligatorio aceptar, mostrando de nuevo el botón para aceptar las condiciones
  - Si no marca ningún *RadioButton* o no pulsa el botón aceptar porque usa el botón atrás del terminal, mostrará el mismo mensaje que si no aceptara
- Además del mensaje en el `EditText` mostrará con `Toast` qué *RadioButton* marcó (o ninguno)
- Cuando se acepten las condiciones desaparecerá el botón de aceptar las condiciones

Ejercicio 3 (2)



Comunicación entre actividades sin usar Intent (datos globales)

1. Usando una clase *Singleton*
2. Usando una clase *Application*

Comunicación entre actividades usando una clase *Singleton*

Definición de clase

```
public class DatosGlobales {
   private static DatosGlobales instancia;
   private String dato;

   private DatosGlobales() {dato="";}

   public static DatosGlobales getInstance() {
      if (instancia == null) instancia = new DatosGlobales();
      return instancia;
   }

  public String getDato() { return dato; }

  public String setDato(String dato) { this.dato=dato; }
}
```

Uso en las actividades

```
DatosGlobales.getInstance().setDato("Valor");

String dato = DatosGlobales.getInstance().getDato();
```

Comunicación entre actividades usando una clase Application (1)

- La clase *Application* es la clase base que representa toda la aplicación
- Gestiona el ciclo de vida global de la app
- Permite almacenar datos y lógica que deben estar disponibles para todas las actividades y componentes de la aplicación
- Debe definirse en `AndroidManifest.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <application
        android:name=".MiApp"
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        ...
```

Comunicación entre actividades usando una clase Application (2)

Definición de clase

```
public class Aplicacion extends MiApp {
   public String dato;
   @Override
   public void onCreate() {
      super.onCreate();
      dato ="Valor inicial";
   }
}
```

Uso en las actividades

```
Aplicacion app =  (Aplicacion) getApplicationContext();
app.dato="Otro valor";
```

Ejercicio 4

Modificar la aplicación de registro anterior de forma que, cuando se hayan aceptado las condiciones, si el usuario vuelve atrás (con el botón físico), el formulario aparezca vacío

Pista: implementa el método *onResume()* en la actividad principal para detectar que se ha vuelto a ella

Actividad 2

Actividad 1

Persistencia de datos (1)

- Android dispone de varias maneras de almacenar datos, la más sencilla la ofrece la API *SharedPreferences* que permite, a aplicaciones y actividades, guardar datos de forma permanente en la forma de parejas clave/valor.
- Su objetivo es almacenar información de usuario, aspectos de configuración, etc. que deban ser mantenidos, aunque se cierre la aplicación.
- Android almacena estos datos en un fichero dentro del directorio **`/data/data/{paquete de la aplicación}/shared_prefs`**
- Los datos de *SharedPreferences* son específicos de la aplicación y se borran al desinstalarla o desde las opciones de Android:

  Configuración/Apps y notificaciones/la aplicación/Almacenamiento/Borrar datos
- El acceso a las *SharedPreferences* puede ser a nivel de actividad o nivel de aplicación:
  - `getPreferences()` obtiene acceso a las preferencias de una actividad.
  - `getSharedPreferences()` obtiene acceso a las preferencias de la aplicación. Puede usarlo todas las actividades. Se debe proporcionar un nombre de fichero.
- Más información:

  https://developer.android.com/training/data-storage/shared-preferences

Persistencia de datos (2)

Guardar datos:

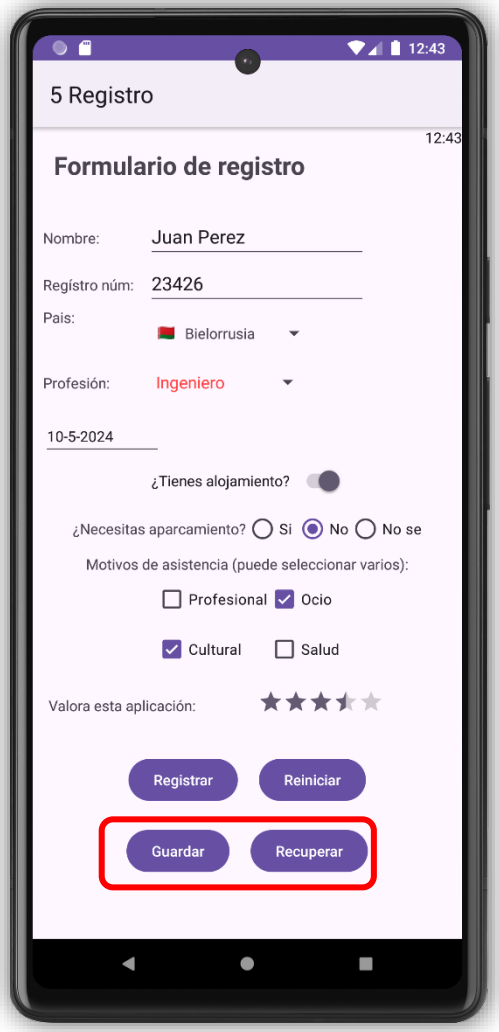
```
final SharedPreferences almacen
      = this.getPreferences(Context.MODE_PRIVATE); // De la actividad actual, no se indica nombre
      = this.getSharedPreferences("Datos.txt",Context.MODE_PRIVATE); // Cualquier actividad de la aplicación
final SharedPreferences.Editor editor = almacen.edit();
editor.putString("clave1", "valor1");
editor.putInt("clave2", 2);
editor.commit();
```

Recuperar datos:

```
String parametro1;
int parametro2;
final SharedPreferences almacen
      = this.getPreferences(Context.MODE_PRIVATE); // De la actividad actual, no se indica nombre
      = this.getSharedPreferences("Datos",Context.MODE_PRIVATE); // Cualquier actividad de la aplicación
if (almacen.contains("clave1")) {
   parametro1 = almacen.getString("clave1","");
}
if (almacen.contains("clave2")) {
   parametro2 = almacen.getInt("clave2",0);
}
```

Ejercicio 5

- Añade a la aplicación de registro anterior dos botones que permitan guardar los datos actuales o recuperar datos guardados, usando la API **`SharedPreferences`**.
- Guarda los datos y busca el fichero donde se han guardado estos en el dispositivo*
- Restricciones:
  - El botón *Recuperar* solo debe mostrarse si al arrancar la aplicación hay datos
  - Si se finaliza el registro, tras aceptar las condiciones deben borrarse los datos guardados en **`SharedPreferences`**



* En el emulador usa *Device File Explorer*. Consulta: https://developer.android.com/studio/debug/device-file-explorer