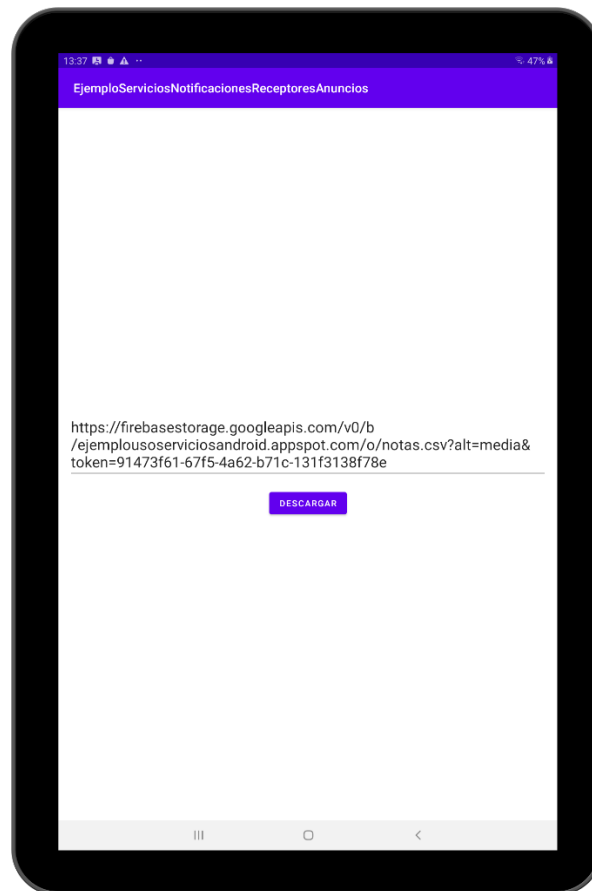


PMDM  
EJEMPLO SERVICIOS, NOTIFICACIONES Y RECEPTORES DE ANUNCIOS

- 1) Crear una app con una pantalla inicial como esta:



En el EditText aparecerá una URL que corresponde a la url de un fichero de texto en formato csv. Es esta url:

<https://firebasestorage.googleapis.com/v0/b/ejemplousoserviciosandroid.appspot.com/o/notas.csv?alt=media&token=91473f61-67f5-4a62-b71c-131f3138f78e>

Ese fichero tiene el siguiente formato:

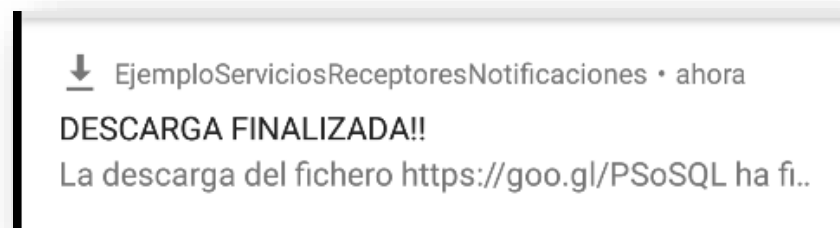
Nombre\_alumno, nombre\_asignatura, nota, profesor

**Ejemplo de contenido de ese fichero:**

PEDRO,MATEMATICAS,5,LUIS LUCIA, LENGUA,1,ANDRES ANABEL,QUIMICA,10,LUCAS ANGEL,FISICA,9,MARIA BELARMINO,MUSICA,2,MARTA CARLA,MATEMATICAS,3.5,LUIS DAMIAN,MATEMATICAS,8,LUIS
--

Cuando pulsamos el botón descargar fichero, **arranca un Servicio en un hilo diferente del UI (con IntentService) que nos descarga el fichero.**

- Cuando la descarga termina, el servicio avisa a la Activity mediante un anuncio de FIN DE DESCARGA y le pasa los datos que contenía el fichero.
  - Si hay problemas de conexión lanza un anuncio de PROBLEMAS CONEXIÓN.
  - Si hay problemas leyendo el fichero lanza un anuncio de PROBLEMAS I/O.
- Justo cuando el receptor de anuncios se entera que la descarga ha finalizado, lanza una **NOTIFICACIÓN** que informa que la descarga ya finalizó y por tanto, el fichero está listo para usarse.

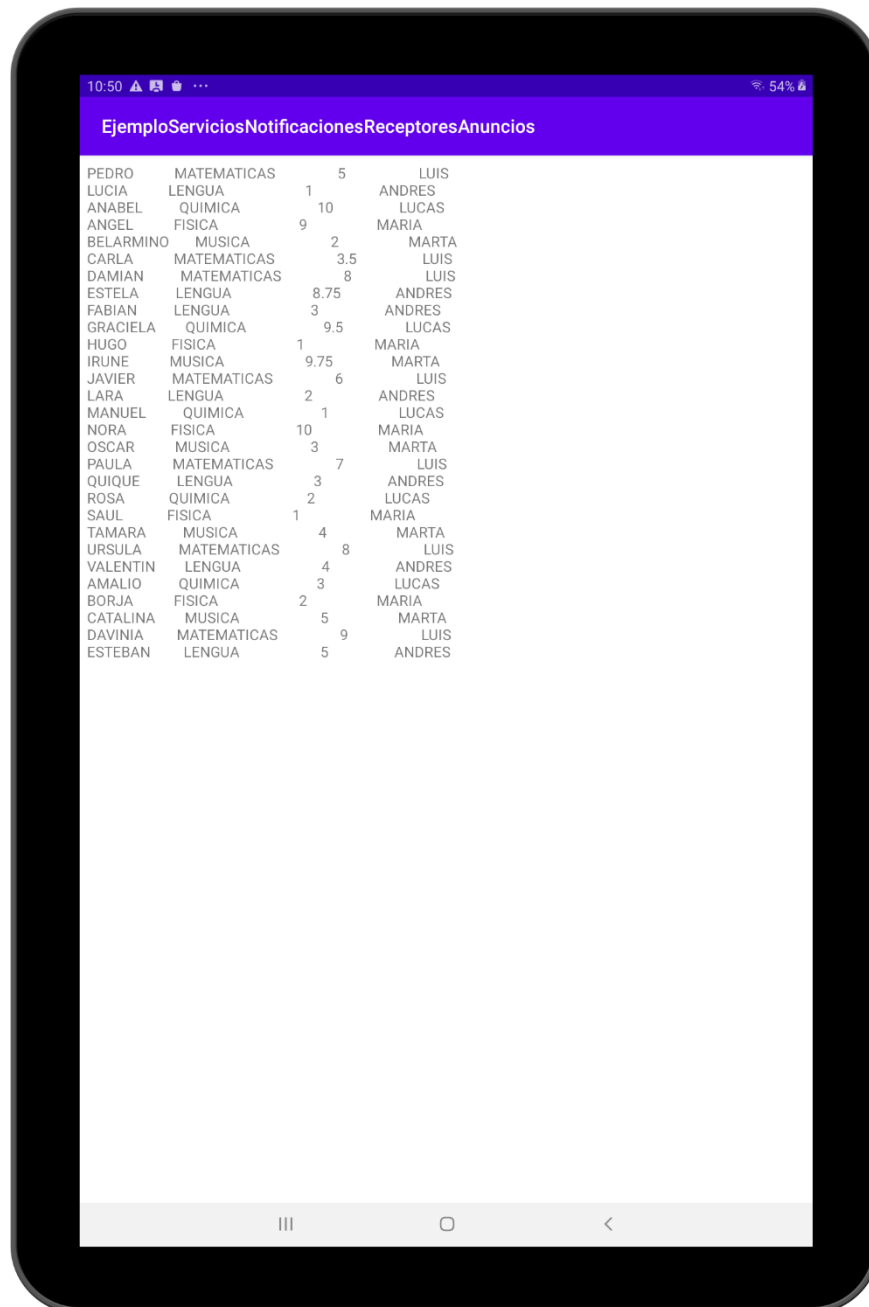


Dicha notificación incluye:

- Una imagen de fin de descarga: `android.R.drawable.stat_sys_download_done`
- Título: "DESCARGA FINALIZADA!!"
- Hora de la notificación
- Texto: "La descarga del fichero .... (poner ahí la url) ha finalizado. Pulse para verlo".

**Si se pulsa sobre dicha notificación, se muestra una activity como la siguiente,** donde se ve el contenido de ese fichero en un TextView de varias líneas y sustituyendo las “,” por espacios en blanco.

PMDM  
EJEMPLO SERVICIOS, NOTIFICACIONES Y RECEPTORES DE ANUNCIOS



## SOLUCIÓN:

**PASO 1:** Diseñamos la pantalla principal y programamos el botón DESCARGAR que nos descargue el fichero (controlar errores de URL mal escrita o problemas de I/O) y nos muestre un mensaje en consola avisando de la descarga y del contenido descargado.

## DISEÑO LAYOUT activity\_main.xml:

```
<androidx.constraintlayout.widget.ConstraintLayout
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"
tools:context=".MainActivity">

    <EditText
        android:id="@+id/etUrlFichero"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:ems="10"
        android:inputType="textUri|textMultiLine"
        android:maxLines="3"
        android:text="@string/url"
        app:layout_constraintBottom_toTopOf="@+id/btDescargar"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainStyle="packed" />

    <Button
        android:id="@+id/btDescargar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/btDescargar_texto"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etUrlFichero"
        app:layout_constraintVertical_chainStyle="packed" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

## Fichero strings.xml:

```
<resources>
    <string name="app_name">EjemploServiciosReceptoresNotificaciones</string>
    <string
name="url">https://drive.google.com/file/d/0BxPB2y1K529TbV1jVFByR3dQVGs/view?usp
=sharing&resourcekey=0-81fykf1L6mCqWlGLGFUQrg</string>
    <string name="btDescargar_texto">DESCARGAR</string>
</resources>
```

**Clase MainActivity.java:**

```
public class MainActivity extends AppCompatActivity {
    // UI
    private EditText etUrlDescarga;
    private Button btDescargar;

    //region MÉTODOS CICLO VIDA ACTIVITY

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

        initReferences();
        setListenersToButtons();
    }
    //endregion

    //region MÉTODOS PROPIOS
    /** Método que obtiene las referencias a las vistas XML
     *
     */
    private void initReferences() {
        etUrlDescarga=findViewById(R.id.etUrlFichero);
        btDescargar=findViewById(R.id.btDescargar);
    }

    /** Método que asigna el escuchador al botón
     *
     */
    private void setListenersToButtons() {
        btDescargar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if(!TextUtils.isEmpty(etUrlDescarga.getText())){
                    lanzarServicio(etUrlDescarga.getText().toString());
                }
            }
        });
    }
    //endregion
}
```

Pero, ¿qué Código añadido en “lanzarServicio()”? ¿Cómo se lanza un servicio? RESPUESTA: simplemente se crea un intent que arranque el servicio; el cual será una clase Java que tenemos que crear y que extiende de IntentService.

Nuestro MainActivity queda entonces:

```
public class MainActivity extends AppCompatActivity {

    // CONSTANTES
    public static final String EXTRA_URI_CONEXION = "uri_descarga";
```

```
// UI
private EditText etUrlDescarga;
private Button btDescargar;

//region MÉTODOS CICLO VIDA ACTIVITY

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

    initReferences();
    setListenersToButtons();
}
//endregion

//region MÉTODOS PROPIOS

/**
 * Método que obtiene las referencias a las vistas XML
 */

private void initReferences() {
    etUrlDescarga = findViewById(R.id.etUrlFichero);
    btDescargar = findViewById(R.id.btDescargar);
}

/**
 * Método que asigna el escuchador al botón
 */
private void setListenersToButtons() {
    btDescargar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (!TextUtils.isEmpty(etUrlDescarga.getText())) {
                lanzarServicio(etUrlDescarga.getText().toString());
            }
        }
    });
}

/**
 * Método que lanza el servicio que descarga el CSV de la url que se le pasa
 *
 * @param urlDescarga url en formato string donde se encuentra el fichero a
descargar
 */

private void lanzarServicio(String urlDescarga) {
    //arranco el servicio, pasándole la url de descarga
    Intent svc = new Intent(this, ServicioLeerArchivoCSV.class);
    svc.putExtra(EXTRA_URI_CONEXION, urlDescarga);
    startService(svc);
}

//endregion
}
```

Ahora tenemos que crear esa clase “**ServicioLeerArchivoCSV.java**” que será el servicio:

Clase ServicioLeerArchivoCSV.java: es el servicio que descarga datos

```
public class ServicioLeerArchivoCSV extends IntentService {
    // CONSTANTES
    public static final String EXTRA_DATOS_CARGADOS = "datos_cargados";
    public static final String ACTION_FIN_CARGA_DATOS =
"es.alejandra.android.ejemploserviciosreceptoresnotificaciones.action.FIN_CARGA_DATOS";
    public static final String ACTION_ERROR_URL =
"es.alejandra.android.ejemploserviciosreceptoresnotificaciones.action.ERROR_URL";
    public static final String ACTION_ERROR_IO =
"es.alejandra.android.ejemploserviciosreceptoresnotificaciones.action.ERROR_IO";

    // URL desde donde descarga
    private URL url;

    // String[] donde guarda los datos descargados
    private String[] notaAlumnoAsignatura;

    public ServicioLeerArchivoCSV() {
        super("ServicioLeerArchivoCSV");
    }

    @Override
    protected void onHandleIntent(@Nullable Intent intent) {
        if (intent != null) {
            if (intent.hasExtra(MainActivity.EXTRA_URI_CONEXION)) {

                try {
                    //recogo la URL para descargar el archivo
                    url = new URL(intent.getStringExtra(MainActivity.EXTRA_URI_CONEXION));
                    // leo el fichero
                    notaAlumnoAsignatura =
                        NetworkUtilities.getResponseFromHttpUrl(url);
                    avisarEventoOcurrido(ACTION_FIN_CARGA_DATOS);
                } catch (MalformedURLException e) {
                    e.printStackTrace();
                    avisarEventoOcurrido(ACTION_ERROR_URL);
                } catch (IOException e) {
                    e.printStackTrace();
                    avisarEventoOcurrido(ACTION_ERROR_IO);
                }
            }
        }
    }
}
```

```
/**
 * Método que lanza un aviso de un evento ocurrido.
 * El evento se le pasa por parámetro
 */
private void avisarEventoOcurrido(String action) {
    Intent iAviso = new Intent(action);
    if (action == ACTION_FIN_CARGA_DATOS) {
        iAviso.putExtra(EXTRA_DATOS_CARGADOS, notaAlumnoAsignatura);
    }
    //EN LOS DEMÁS CASOS, NO LLEVA EXTRAS, POR ESO NO SE AÑADEN
    sendBroadcast(iAviso);

    Log.d("MIAPLI", action);
}
}
```

Acordarse de pedir en el manifest, el permis de Internet:

```
<uses-permission android:name="android.permission.INTERNET" />
```

Revisar que en el **AndroidManifest** nos haya registrado nuestro servicio:

```
<service android:name=".ServicioLeerArchivoCSV"></service>
```

Y la clase de ayuda que uso para leer el fichero **NetworkUtilities.java** es:

```
public final class NetworkUtilities {

    /**
     * This method returns the entire result from the HTTP response as an array of
     * Strings (cada linea del fichero).
     *
     * @param url The URL to fetch the HTTP response from.
     * @return The contents of the HTTP response.
     * @throws IOException Related to network and stream reading
     */
    public static String[] getResponseFromHttpUrl(URL url) throws IOException {
        BufferedReader br;
        ArrayList<String> lineasList = new ArrayList<>();
        String linea;

        HttpURLConnection urlConnection = (HttpURLConnection) url.openConnection();

        try {
            br = new BufferedReader(new
            InputStreamReader(urlConnection.getInputStream()));

            while ((linea = br.readLine()) != null) {
                lineasList.add(linea);
            }
            br.close();
            if (lineasList.size() > 0)
                return (lineasList.toArray(new String[lineasList.size()]));
        }
    }
}
```



```
        else
            return null;
    } finally {
        urlConnection.disconnect();
    }
}
```

**PASO 2:** Quitamos los mensajes de log que pusimos en el servicio, si queremos, y en el MAINACTIVITY creamos un **RECEPTOR DE ANUNCIOS** que recoja los anuncios que envíe el Servicio. Para ello:

- Definimos una clase `ReceptorAvisosServicio`
- Registramos el receptor de anuncios en el `onResume()`
- Lo desregistramos en el `onPause()`.

Nos queda el `MainActivity`:

```
public class MainActivity extends AppCompatActivity {
    // CONSTANTES
    public static final String EXTRA_URI_CONEXION = "uri_descarga";
    public static final String TAG_APLI = MainActivity.class.getSimpleName();

    // UI
    private EditText etUrlDescarga;
    private Button btDescargar;

    //RECEPTOR AVISOS SERVICIO
    private ReceptorAvisosServicio receptorAvisosServicio;

    //GESTION DATOS FICHERO
    String[] datosAlumnosNotas;

    //region MÉTODOS CICLO VIDA ACTIVITY

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

        initReferences();
        setListenersToButtons();
    }

    @Override
    protected void onResume() {
        super.onResume();

        //Registro el receptor de Broadcast (BroadcastReceiver) para que la activity escuche los eventos que
        //le envía
        //nuestro servicio
        receptorAvisosServicio = new ReceptorAvisosServicio();
        IntentFilter intentFilter = new IntentFilter();
        intentFilter.addAction(ServicioLeerArchivoCSV.ACTION_FIN_CARGA_DATOS);
        intentFilter.addAction(ServicioLeerArchivoCSV.ACTION_ERROR_IO);
        intentFilter.addAction(ServicioLeerArchivoCSV.ACTION_ERROR_URL);
        registerReceiver(receptorAvisosServicio, intentFilter);
    }
}
```

```

@Override
protected void onPause() {
    super.onPause();
    //desregistro el BroadcastReceiver
    unregisterReceiver(receptorAvisosServicio);
}

//endregion

//region MÉTODOS PROPIOS

/**
 * Método que obtiene las referencias a las vistas XML
 */
private void initReferences() {
    etUrlDescarga = findViewById(R.id.etUrlFichero);
    btDescargar = findViewById(R.id.btDescargar);
}

/**
 * Método asigna el escuchador al botón
 */
private void setListenerToButtons() {
    btDescargar.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (!TextUtils.isEmpty(etUrlDescarga.getText())) {
                lanzarServicio(etUrlDescarga.getText().toString());
            }
        }
    });
}

/**
 * Método que lanza el servicio que descarga el CSV de la url que se le pasa
 *
 * @param urlDescarga url en formato string donde se encuentra el fichero a descargar
 */
private void lanzarServicio(String urlDescarga) {
    //arranco el servicio, pasándole la url de descarga
    Intent svc = new Intent(this, ServicioLeerArchivoCSV.class);
    svc.putExtra(EXTRA_URI_CONEXION, urlDescarga);
    startService(svc);
}

/**
 * Método que muestra en el LogCat los datos que se le pasan
 *
 * @param datosAlumnosNotas String[] a mostrar en el log
 */
private void mostrar_en_log(String[] datosAlumnosNotas) {
    for (String asignaturaAlumno : datosAlumnosNotas) {
        Log.d("MIAPLI", asignaturaAlumno);
    }
}

//endregion

```

PMDM  
EJEMPLO SERVICIOS, NOTIFICACIONES Y RECEPTORES DE ANUNCIOS

```
//region RECEPTOR DE ANUNCIOS
```

```
RECEPTOR AVISOS SERVICIO
```

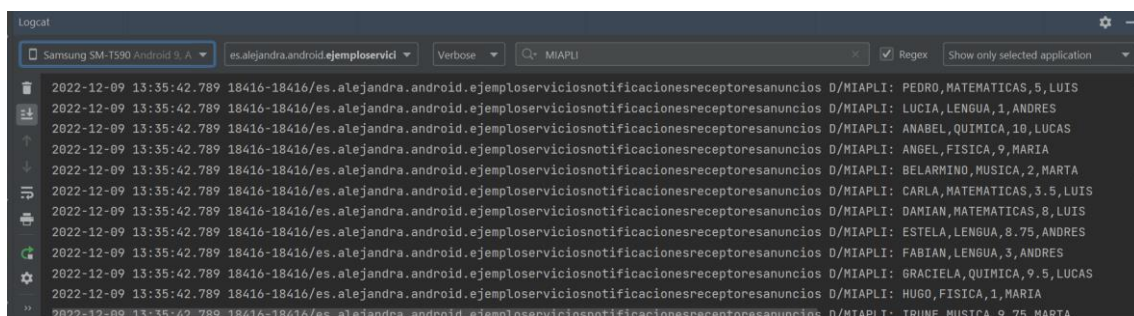
```
public class ReceptorAvisosServicio extends BroadcastReceiver {

    public ReceptorAvisosServicio() {

    }

    @Override
    public void onReceive(Context context, Intent intent) {
        switch (intent.getAction()) {
            case ServicioLeerArchivoCSV.ACTION_FIN_CARGA_DATOS:
                datosAlumnosNotas =
intent.getStringArrayExtra(ServicioLeerArchivoCSV.EXTRA_DATOS_CARGADOS);
                mostrar_en_log(datosAlumnosNotas);
                Toast.makeText(context, "FIN DESCARGA", Toast.LENGTH_SHORT).show();
                return;
            case ServicioLeerArchivoCSV.ACTION_ERROR_URL:
                Toast.makeText(context, "URL INCORRECTA", Toast.LENGTH_SHORT).show();
                return;
            case ServicioLeerArchivoCSV.ACTION_ERROR_IO:
                Toast.makeText(context, "ERROR DE LECTURA EN EL FICHERO",
Toast.LENGTH_SHORT).show();
                return;
        }
    }
}
```

Si lo ejecutamos ahora, deberíamos ver en el LogCat el contenido del fichero.



```
Logcat
Samsung SM-T590 Android 9.0
es.alejandra.android.ejemploservicio
Verbose
MIAPLI
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: PEDRO, MATEMATICAS, 5, LUIS
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: LUCIA, LENGUA, 1, ANDRES
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: ANABEL, QUIMICA, 10, LUCAS
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: ANGEL, FISICA, 9, MARIA
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: BELARMINO, MUSICA, 2, MARTA
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: CARLA, MATEMATICAS, 3.5, LUIS
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: DAMIAN, MATEMATICAS, 8, LUIS
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: ESTELA, LENGUA, 8.75, ANDRES
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: FABIAN, LENGUA, 3, ANDRES
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: GRACIELA, QUIMICA, 9.5, LUCAS
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: HUGO, FISICA, 1, MARIA
2022-12-09 13:35:42.789 18416-18416/es.alejandra.android.ejemploserviciosnotificacionesreceptoresanuncios D/MIAPLI: TRINIDAD, MUSICA, 9.75, MARTA
```

**PASO 3:** Programo la notificación que se debe mostrar cuando finaliza la descarga **dentro del Receptor de anuncios**, puesto que dice que cuando éste se entera de que la descarga finalizó, la lanza. En realidad, las notificaciones se pueden crear en una Activity, Servicio u otro componente; pero aquí la necesitamos en el Receptor de anuncios.

**De momento no sucederá nada al pulsar sobre la notificación. Simplemente la veré en la barra de estado y la puedo desplegar.** Por tanto, dentro del Receptor de anuncios añado:

La clase **ReceptorAvisosServicio.java** queda así:

```
//region RECEPTOR DE ANUNCIOS
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
RECEPTOR  AVISOS  SERVICIO  //////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
public class ReceptorAvisosServicio extends BroadcastReceiver {

    // CONSTANTES
    static final String EXTRA_KEY_DATOS_FICHERO = "extra_key_fichero";
    static final int COD_REQUEST_PENDING_INTENT = 0;

    // NOTIFICACIÓN
    private NotificationManager notificationManager;
    static final String CANAL_ID = "mi_canal";
    static final int NOTIFICACION_ID = 1;

    public ReceptorAvisosServicio() {

    }

    @Override
    public void onReceive(Context context, Intent intent) {
        switch (intent.getAction()) {
            case ServicioLeerArchivoCSV.ACTION_FIN_CARGA_DATOS:
                datosAlumnosNotas =
intent.getStringArrayExtra(ServicioLeerArchivoCSV.EXTRA_DATOS_CARGADOS);
                mostrar en log(datosAlumnosNotas);
                //lanzo la notificación
                lanzarNotificacion(datosAlumnosNotas);
                return;
            case ServicioLeerArchivoCSV.ACTION_ERROR_IO:
                Toast.makeText(context, "ERROR DE LECTURA EN EL FICHERO",
Toast.LENGTH_SHORT).show();
                return;
            case ServicioLeerArchivoCSV.ACTION_ERROR_URL:
                Toast.makeText(context, "ERROR URL INCORRECTA",
Toast.LENGTH_SHORT).show();
                return;
        }
    }

    /** Método que crea y lanza la notificación
     *
     * @param datos el contenido del fichero descargado en formato String[]
     */
    private void lanzarNotificacion(String[] datos) {

        notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            // Sería para versiones Oreo o superiores

            //creo un canal de notificaciones
            crearCanalNotificacion(notificationManager);
        }
    }
}
```

```
//creo la notificación
Notification notificacion = crearNotificacion(notificationManager);
// lanzo la notificación
notificationManager.notify(NOTIFICACION_ID, notificacion);
}
```

Y los métodos usados dentro de lanzarNotificación() y que permiten crear el canal de la notificación y luego la notificación son:

#### Método crearCanalNotificacion(...):

```
/**
 * Método que crea el canal que voy a usar para las notificaciones de mi app
 *
 * @param notificationManager es el servicio del sistema Android que gestiona y ejecuta
las notificaciones
 */
@RequiresApi(api = Build.VERSION_CODES.O)
private void crearCanalNotificacion(NotificationManager notificationManager) {
    //creo un canal de notificaciones
    NotificationChannel notificationChannel =
        new NotificationChannel(CANAL_ID, "Mis notificaciones",
                                NotificationManager.IMPORTANCE_DEFAULT);
    notificationChannel.setDescription("Es el canal empleado para notificar el servicio
de descarga");
    notificationManager.createNotificationChannel(notificationChannel);
}
```

#### Método crearCanalNotificacion(...):

```
/**
 * Método que crea la notificación
 *
 * @param notificationManager es el servicio del sistema Android que gestiona y ejecuta
las notificaciones
 */
private Notification crearNotificacion(NotificationManager notificationManager) {
    NotificationCompat.Builder notificacion =
        new NotificationCompat.Builder(getApplicationContext(), CANAL_ID)
            .setSmallIcon(android.R.drawable.stat_sys_download_done)
            .setContentTitle("DESCARGA FINALIZADA!!")
            .setWhen(Calendar.getInstance().getTimeInMillis())
            .setAutoCancel(true)
            .setContentText("La descarga del fichero " +
                etUrlDescarga.getText().toString() +
                " ha finalizado. Pulse para verlo.");
    return notificacion.build();
}
```

**PASO 6:** Vamos a añadirle ahora funcionalidad a la notificación para que al pulsar sobre ella, nos lleve a otra activity que muestre en un TextView el contenido del fichero descargado cambiando las “,” por tabuladores. Para ello, primero necesito crear la otra activity que llamaré: **MostrarFicheroActivity**:

Su **layout** será:

```
<androidx.constraintlayout.widget.ConstraintLayout
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"
tools:context=".MostrarFicheroActivity">

    <TextView
        android:id="@+id/tvContenidoFichero"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:textSize="16sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Y su java **MostrarFicheroActivity.java** de **momento no lo tocamos**.

Vamos a configurar en el Receptor de anuncios la notificación que hicimos en el punto anterior, para que al pulsar sobre ella, nos lance la activity: “MostrarFicheroActivity”. Para ello, **debemos añadir 2 constantes nuevas en el Receptor de Anuncios y luego modificar el método “lanzarNotificacion” que hay en el receptor de anuncios del MainActivity y el método crearNotificacion(..) porque hay que añadirle a la notificación que cuando se pulse sobre ella lance ese PendingIntent**. Las constantes en el receptor y el método lanzarNotificación quedarán:

```
public class ReceptorAvisosServicio extends BroadcastReceiver {

    // CONSTANTES
    static final int COD_REQUEST_PENDING_INTENT = 0;
    static final String EXTRA_KEY_DATOS_FICHERO = "extra_key_fichero";

    // NOTIFICACIÓN
    private NotificationManager notificationManager;
    static final String CANAL_ID = "mi canal";
    static final int NOTIFICACION_ID = 1;
```

```
public ReceptorAvisosServicio() {
}

@Override
public void onReceive(Context context, Intent intent) {
    switch (intent.getAction()) {
        case ServicioLeerArchivoCSV.ACTION_FIN_CARGA_DATOS:
            datosAlumnosNotas =
intent.getStringArrayExtra(ServicioLeerArchivoCSV.EXTRA_DATOS_CARGADOS);
            mostrar_en_log(datosAlumnosNotas);
            // lanzo la notificación
            lanzarNotificacion(datosAlumnosNotas);
            return;
        case ServicioLeerArchivoCSV.ACTION_ERROR_IO:
            Toast.makeText(context, "ERROR DE LECTURA EN EL FICHERO",
Toast.LENGTH_SHORT).show();
            return;
        case ServicioLeerArchivoCSV.ACTION_ERROR_URL:
            Toast.makeText(context, "ERROR URL INCORRECTA",
Toast.LENGTH_SHORT).show();
            return;
    }
}

/**
 * Método que crea y lanza la notificación
 *
 * @param datos el contenido del fichero descargado en formato String[]
 */
private void lanzarNotificacion(String[] datos) {
    notificationManager =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        // Sería para versiones Oreo o superiores
        crearCanalNotificacion(notificationManager);
    }

    // creo un pendingIntent que será ejecutado al pulsar sobre la notificación y
    // que me lanzará la activity que se indica (se le pasan los datos para que
    // esa nueva activity pueda mostrarlos)
    PendingIntent pendingIntent=crearPendingIntent(datos);

    // creo la notificación
    Notification notificacion = crearNotificacion(notificationManager, pendingIntent);
    // lanzo la notificación
    notificationManager.notify(NOTIFICACION_ID, notificacion);
} // fin método lanzarNotificacion
```

PMDM  
EJEMPLO SERVICIOS, NOTIFICACIONES Y RECEPTORES DE ANUNCIOS

Para crear el PendingIntent, el método `crearPendingIntent(...)` es:

```
/**
 * Método que crea un PendingIntent que se ejecutará al pulsar sobre la notificación y
 * que abrirá una segunda activity para mostrar en ella los datos que se pasan como
 * parámetro.
 *
 * Un PendingIntent es un Intent que especifica una acción que se llevará a cabo en el
 * futuro.
 *
 * @param datos son los datos a mostrar en la otra activity (el fichero descargado)
 * @return el PendingIntent creado
 */
private PendingIntent crearPendingIntent(String[] datos) {
    // Intent que quiero que se ejecute en el futuro (al pulsar sobre la notificación
    // en este caso)
    Intent iMostrarFichero = new Intent(getApplicationContext(),
                                           MostrarFicheroActivity.class);
    iMostrarFichero.putExtra(EXTRA_KEY_DATOS_FICHERO, datos);
    // creo el PendingIntent
    PendingIntent intentPendiente = PendingIntent.getActivity(MainActivity.this,
                                                             COD_REQUEST_PENDING_INTENT,
                                                             iMostrarFichero, 0); //flags=0; es sin flags, de ahí ese 0
    return intentPendiente;
}
```

La modificación que hay que hacer en el método `crearNotificacion(...)` es:

```
/**
 * Método que crea la notificación
 *
 * @param notificationManager es el servicio del sistema Android que gestiona y ejecuta
 * las notificaciones
 */
private Notification crearNotificacion(NotificationManager notificationManager ,
                                       PendingIntent intentPendiente)
{
    NotificationCompat.Builder notificacion =
        new NotificationCompat.Builder(getApplicationContext(), CANAL_ID)
            .setSmallIcon(android.R.drawable.stat_sys_download_done)
            .setContentTitle("DESCARGA FINALIZADA!!")
            .setWhen(Calendar.getInstance().getTimeInMillis())
            .setAutoCancel(true)
            .setContentText("La descarga del fichero " +
                           etUrlDescarga.getText().toString() +
                           " ha finalizado. Pulse para verlo.")
            .setContentIntent(intentPendiente);

    return notificacion.build();
}
```

Ahora ya podemos modificar el java `MostrarFicheroActivity.java` para que recoja el `String[]` que le estamos pasando con el intent del PendingIntent en la notificación y lo muestre en el TextView. Se hace como si fuese enviado con un intent normal, es decir, con `getIntent().getExtras().....`

Nos quedará:



```
public class MostrarFicheroActivity extends AppCompatActivity {

    private String[] datosFichero;
    // UI
    private TextView tvTextoFichero;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_mostrar_fichero);
        initReferences();
        // recojo el texto del fichero del intent
        if (getIntent()
            .hasExtra(MainActivity.ReceptorAvisosServicio.EXTRA_KEY_DATOS_FICHERO))
        {
            datosFichero = getIntent().getStringArrayExtra(
                MainActivity.ReceptorAvisosServicio.EXTRA_KEY_DATOS_FICHERO);
        }

        if (datosFichero != null && datosFichero.length > 0) {
            // hay datos
            for (String linea : datosFichero) {
                //extraigo cada palabra de la línea (el nombre alumno,
                // su asignatura,...)
                String[] lineaTroceada = linea.split(",");
                // añadido al TextView la línea formateada con espacios entre palabras
                tvTextoFichero.append(String.format("%-16s", lineaTroceada[0]) +
                    String.format("%-28s", lineaTroceada[1]) +
                    String.format("%-15s", lineaTroceada[2]) +
                    String.format("%10s", lineaTroceada[3]) + "\n");
                // %-10s --> el "-" indica que rellene por la derecha con espacios
                // blancos
            }
        }

        /**
         * Método que obtiene las referencias a las vistas XML
         */
        private void initReferences() {
            tvTextoFichero = findViewById(R.id.tvContenidoFichero);
        }
    }
}
```