DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

Grado en Ingeniería Informática en Tecnologías de la Información – 2020-21

Práctica 2

MisPelis

1. Objetivos fundamentales:

- Avanzar en el conocimiento del manejo de layouts avanzados.
- Aprender los conceptos básicos del almacenamiento persistente de ficheros (interno / externo) en la plataforma Android.
- Manejar el uso de SQLite como gestor de base de datos relacional en las plataformas.
- La práctica se entregará a través Campus virtual de la asignatura en la sección prácticas. Se deberá entregar en un archivo comprimido. Todos aquellos archivos que sean detectados como virus no serán corregidos. La fecha límite de entrega será el **12 de mayo del 2021 a las 23:55h**.

2. Descripción de la práctica.

PARTE I (8 puntos)

La idea de esta práctica es construir una aplicación para almacenar películas de diferentes plataformas como NETFLIX, HBO, etc. Como base de la práctica utilizaremos la práctica 1, de la que ya tenemos implementado la parte gráfica y el comportamiento del Login del aplicativo. Lo que deberemos implementar añadido a esto es la persistencia de usuarios en la base de datos SQLite.

Usuarios: Se deberá almacenar en SQLite toda la información relativa a los usuarios que se especificaba en la práctica 1.

Plataformas: Deberemos almacenar las diferentes plataformas de las cuales queramos almacenar películas. La información para guardar en SQLite será:

- Id: identificador de la plataforma (autogenerado, no visualizable ni editable).
- Id de usuario → relacionado con la tabla de usuarios
- Imagen: Imagen representativa de la plataforma.
- Nombre de la plataforma: (Netflix, HBO, etc…)

- URL de acceso a la plataforma:
- Usuario de acceso a la plataforma (correo, dni o similar)
- Password de acceso a la plataforma

Películas:

- Id: identificador de la película (autogenerado, no visualizable ni editable).
- Id de usuario → Relacionado con la tabla de usuarios
- Id de la Plataforma → Relacionado con la tabla de Plataformas
- Carátula: Imagen representativa de la película.
- Título de la película
- Duración en minutos
- Género cinematográfico: Drama, thriller, etc…
- Calificación de 5 a 5 en función del propio usuario.

Una vez realizado el Login en el aplicativo se mostrará una pantalla donde se podrá seleccionar entre Plataformas y películas. Cada uno de estos botones / imágenes llevará a otra pantalla de listado donde se mostrarán (filtrados por el usuario que ha entrado) cada uno de los registros de las diferentes entidades (plataformas / películas) listados por orden alfabético, se visualizará imagen y texto.

Evidentemente, al inicio no existirá ninguna plataforma ni película y será necesario darlos de alta. Para ello podremos optar por utilizar un menú en la parte superior o bien un botón físico en la pantalla. Al pulsar la opción de alta de servicio se mostrará en una nueva actividad de alta / edición con los campos para dar de alta. Además de este botón de alta existirá otro botón de exportar que lo que hará es exportar toda la información del listado en un fichero físico en el almacenamiento externo de Android.

LISTADO DE PANTALLAS

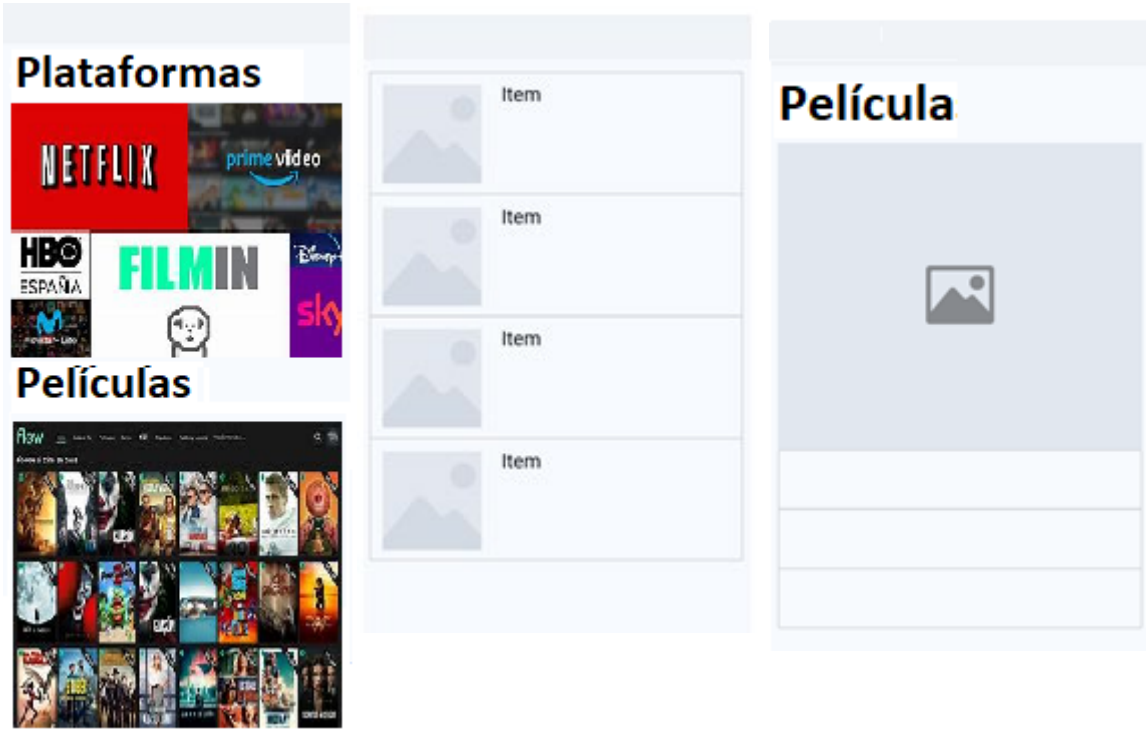
- Pantallas heredadas de la práctica 1
- Pantalla inicial con selección de Plataformas y películas

- Plataformas
  - Pantalla con listado de plataformas
  - Pantalla con visualización de plataforma

        - Contiene a su vez un listado con las películas asociadas a esa plataforma
    - Pantalla con edición de plataforma

- Películas
    -
    - Pantalla con listado de películas
    - Pantalla con visualización de película
    - Pantalla con edición de película

Ejemplo de visualización de pantallas



COMPORTAMIENTO EN PANTALLAS DE LISTADOS

En el caso de los listados de plataformas y películas se deberá implementar el siguiente comportamiento:

- La primera vez que arranquemos la aplicación no existirá ningún servicio ni ningún gasto, en este caso mostraremos un Toast indicando que no hay ningún elemento grabado de ese tipo.
- Hay que implementar la posibilidad de que cuando el usuario realice una pulsación corta sobre un ítem de la lista, se mostrará el detalle del elemento con los datos del registro correspondiente y esta pantalla tendrá un menú superior con la opción de editar para permitir la edición de los datos.

- Si el usuario realiza una pulsación prolongada sobre el ítem de la lista, aparecerá la opción de eliminar dicho registro mediante un menú contextual al elemento seleccionado. Habrá que seguir los preceptos de usabilidad y ante una opción de borrado, preguntar al usuario si desea realmente realizar dicha operación.
- Para asignar imágenes se podrá hacer bien haciendo una foto con el móvil o bien seleccionándola de la galería de imágenes.

ALMACENAMIENTO DE LOS DATOS

Como hemos comentado, para salvaguardar los datos de usuarios, plataformas y películas deberemos utilizar una base de datos SQLite llamada BDPelis.db y acceder a su información usando las funciones vistas en clase durante el curso.

Existirán 3 tablas: usuarios, plataformas y películas utilizando los tipos de datos que mejor se adapten a cada uno de los campos.

Las fotos de los plataformas y películas se almacenarán también en el almacenamiento del dispositivo, pudiendo elegir entre hacerlo en la propia base de datos o en el almacenamiento interno del dispositivo.

Además, hay que recordar que en las pantallas de listados se deberá implementar la posibilidad de exportar la información del listado (solo datos no fotos) en un fichero de texto plano con el formato que se crea conveniente: csv, xml, json o similar.

Recomendaciones y cosas a tener en cuenta

Las maquetas de las pantallas es meramente indicativa, la apariencia se puede cambiar y se puede utilizar cualquier componente siempre que tenga sentido con la representación. Se valorará positivamente el diseño alternativo de las pantallas.

Se recomienda abordar la práctica en pequeños trozos agrupados por funcionalidades. Por ejemplo, empezar por implementar los layouts y su comportamiento. Una vez tengo eso implementar la parte del almacenamiento de SQLite para una entidad y en tener esta parte se puede exportar al resto y así

sucesivamente. Asímismo, se recomienda no complicarse mucho la vida con los componentes y layouts. Mejor empezar con interfáz básico y luego ir complicando una vez que tenermos todo lo demás implementado.

Es obligatorio utilizar los ficheros de recursos para layouts, cadenas, imágenes, dimensiones, etc…

Recordar que es necesaria la separación de las diferentes clases utilizando espacios de nombres.

PARTE II (2 puntos)

- Utilización de técnicas avanzadas de visualización de interfaces gráficas.
    - RecyclerView: https://developer.android.com/guide/topics/ui/layout/recyclerview
    - ViewPage: https://developer.android.com/guide/navigation/navigation-swipe-view
    - CardView
    - Animaciones

  Existen multitud de ejemplos que se pueden utilizar como base en la siguiente URL:
    - https://github.com/android/views-widgets-samples
- Utilización de alguna o varias características de la Biblioteca JectPack https://developer.android.com/jetpack como, por ejemplo:
    - CameraX
    - DaraStore
    - Room:https://developer.android.com/training/data-storage/room

3. Evaluación

- **(8 puntos) PARTE I**

  - **(2 puntos) Pantallas de login, plataformas y películas.**
  - **(3 puntos) Almacenamiento de gastos en SQLite.**
  - **(2 puntos) Almacenamiento de imágenes en BD o fichero físico.**
  - **(1 punto) Exportación de listados en fichero en memoria externa.**
- **(2 puntos) PARTE II:**
  - **Utilización de técnicas avanzadas de visualización de interfaces gráficas y utilización de librerías de terceros.**

Normas obligatorias

El nombre del zip en que se entregue el proyecto debe ser obligatoriamente con el siguiente formato "mispelis00000000x" y el paquete raiz será es.umh.dadm. mispelis00000000x. (donde 00000000X será la identificación del alumno: dni, tarjeta de residente, etc… que realiza la práctica)

La práctica hay que hacerla utilizando los siguientes requerimientos:

- Versiones de Android Studio permitidas
  - Android 3.5.3 (máquina virtual)
  - Android 4.1.2 (última versión a día de hoy)
- Android 11 API 30
- Minimun API Level: API 22: Android 5.1 (Lollipop).

En caso de no cumplir estos requerimientos puede conllevar al suspenso de la práctica.

Junto con la práctica hay que realizar una memoria explicativa con los pasos seguidos, la estratégia para abordar la práctica así como indicar en detalle lo que se ha abordado de la PARTE II.

ANEXO I. ALMACENAMIENTO FICHEROS INTERNO y SDCARD

Android nos provee con diversos métodos de almacenamiento persistente de los datos manejados en las aplicaciones. Entre estos métodos destacan los siguientes:

- Manejo de ficheros en la **memoria interna** del dispositivo. Por defecto, estos ficheros son privados a la aplicación que los crea, ninguna otra aplicación tiene acceso a ellos. La desinstalación de la aplicación provoca el borrado de dichos ficheros.
- Manejo de ficheros en la memoria externa del dispositivo. Tales como tarjetas SD, MMC, etc.

Empezaremos practicando sobre el tipo de almacenamiento interno y externo de ficheros.

Poniendo toda nuestra atención en el método de almacenamiento interno, comenzaremos realizando una práctica sencilla en la que comprobaremos dicha funcionalidad.

Partiendo de las dos alternativas que tenemos en almacenamiento interno, la primera con métodos de la librería de I/O de Java, y la segunda, con métodos propios de Android, estudiaremos esta última por su facilidad de uso.

Android para facilitar esta tarea nos provee del método openFileOutput(), que recibe como parámetros el nombre del fichero y el modo de acceso con el que queremos abrirlo. Los diferentes modos de acceso que tiene este método implementado son los siguientes:

Modo	Descripción
MODE_APPEND	Si el fichero existe, añadir contenido al final del mismo. Si no existe, se crea
MODE_PRIVATE	Modo por defecto donde el fichero sólo puede ser accedido por la aplicación que lo crea
MODE_WORLD_READABLE	Permite a las demás aplicaciones tener acceso de lectura al fichero creado
MODE_WORLD_WRITEABLE	Permite a las demás aplicaciones tener acceso de escritura al fichero creado

Utilizando el método OutputStreamWriter nos permitirá escribir una cadena de texto al fichero. El siguiente código nos muestra su uso:

```
try
{
    OutputStreamWriter fout=
        new OutputStreamWriter(
            openFileOutput("fichero_interno.txt",
     Context.MODE_PRIVATE));

    fout.write("Introduccion de texto en el fichero.");
    fout.close();
}
catch (Exception ex)
{
    Log.e("Ficheros", "Error al escribir fichero a memoria
interna");
   }
```

Como se indicó en teoría si deseamos poder visualizar la creación de nuestro archivo, una vez ejecutado el código deberemos activar la perspectiva DDMS y buscar dicho fichero en la ruta siguiente:

```
/data/data/paquete_java/files/nombre_fichero
```

Para leer ficheros tan sólo tendremos que indicar al método openFileInput () que lo vamos a abrir para lectura utilizando los métodos propios de la librería java.io:

```
try
{
    BufferedReader fin =
        new BufferedReader(
            new InputStreamReader(
                openFileInput("prueba_int.txt")));
```

```
            String texto = fin.readLine();
            fin.close();
        }
        catch (Exception ex)
        {
            Log.e("Ficheros", "Error al leer fichero desde memoria
        interna");
              }
```

Pasemos a ver, a continuación, los métodos de acceso de almacenamiento en memoria externa. Una buena práctica de programación para este uso de almacenamiento consiste como primer paso ineludible en comprobar la disponibilidad de dicha memoria externa. Con el método **getExternalStorageState()** de la clase Environment tendremos resuelto el problema. Veremos cómo comprobar dicho estado en el siguiente código:

```
//Creamos dos variables booleanas que nos permitirán registrar los
estados de la memoria
boolean mExternaHabilitada= false;
boolean mExternaEscribible= false;
//Crearemos una string para guardar el estado de dicha memoria
String estadoMemoria = Environment.getExternalStorageState();
//Comprobaremos si podemos leer y escribir en la memoria externa
if(estadoMemoria.equals(Environment.MEDIA_MOUNTED)){
      mExternaHabilitada = true;
      mExternaEscribible = true;
} //Podremos leer pero no escribir
else if (estadoMemoria.equals(Environment.MEDIA_MOUNTED_READ_ONLY)){
            mExternaHabilitada = true;
            mExternaEscribible = false;
else
//No se puede ni leer ni escribir
      {
            mExternaHabilitada = false;
            mExternaEscribible = false;
      }
```

Una vez comprobada la disponibilidad usaremos el método getExternalFilesDir () de la clase Context para abrir el fichero en cuestión. Si deseamos guardar archivos en la

memoria externa que no queremos que sean borrados cuando la aplicación se desinstala, tan sólo deberemos guardar dichos ficheros en los directorios públicos que cuelgan de la memoria externa como pueden ser Music/, Pictures/, Ringtones/, etc.

ANEXO II. SQLite

La base de datos SQLite es un motor de base de datos que cada día tiene mejor aceptación entre la gran comunidad de desarrolladores. Entre sus excelencias, destacan por encima del resto, que no necesita un servidor, su configuración es extremadamente sencilla y simple, y el tamaño es minúsculo. Si a todo eso le añadimos que es código abierto, estamos ante una oferta realmente tentadora a la hora de implementar nuestros desarrollos.

Esta herramienta se basa sobre la clase SQLiteOpenHelper y su función no es otra que la de comportarse como una plantilla sobre la que personalizaremos nuestra base de datos.

Si creamos una clase SQLiteOpenHelper estamos obligados forzosamente a implementar los siguientes métodos:

- onCreate (SQLiteBBDD)
- onUpgrade (SQLiteBBDD, int, int)

onOpen(SQLiteBBDD) está última con carácter opcional.

Deberemos de hacer uso del patrón de diseño visto en clase y crear una clase **Adaptador** en donde se encontrará contenida la clase **SQLiteOpenHelper**. Su función no es otra que la de comportarse como una plantilla sobre la que personalizaremos nuestra base de datos.