# DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

## Grado en Ingeniería Informática en Tecnologías de la Información

## Curso 2022 / 23

**Práctica 1**

**Mis películas**

# 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 de la web de la asignatura en su correspondiente apartado de tareas. Se deberá entregar en un archivo comprimido. Todos aquellos archivos que sean detectados como virus no serán corregidos. La fecha de entrega será el **1 de Mayo del 2023 a las 23:55h. Se podrá entregar en una segunda entrega el 8 de mayo del 2023 (en esta entrega se descontará un punto de la nota final).**

# 2. Descripción de la práctica.

## PARTE I

La idea de esta práctica es construir una aplicación para almacenar películas de las diferentes plataformas como NETFLIX, HBO, etc. Lo primero que se mostrará es una pantalla para hacer Login y en caso de no tener cuenta permitir la creación de una cuenta nueva.

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.

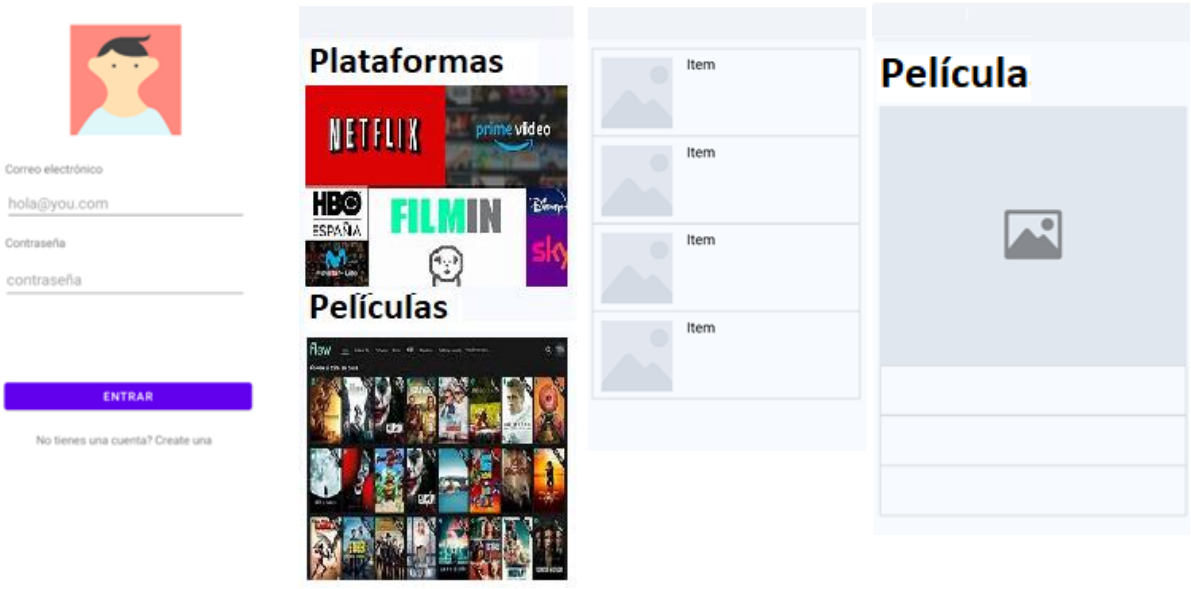
Evidentemente, al inicio no existirá ninguna plataforma ni película y será necesario darlos de alta. Para ello podremos optar por utilizar un menú en la

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

- Pantalla de Login
- Pantalla de registro
- 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 / creación de plataforma
- Películas
  - Pantalla con listado de películas
  - Pantalla con visualización de película
  - Pantalla con edición / creación de película

Ejemplo de visualización de pantallas




Nota: Las maquetas de las pantallas son meramente indicativas, 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.

## COMPORTAMIENTOS EN PANTALLAS DE LOGIN Y REGISTRO

En el caso del registro de se deberá implementar el siguiente comportamiento:

**Pantalla de Login**: Mostrará un formulario con los siguientes campos (ambos obligatorios):

- Email
- Password.

Las acciones a realizar son:

- Entrar: haciendo clic se validará si se han rellenado los datos obligatorios, en caso contrario se mostrará un mensaje de tipo Toast advirtiendo de los campos que no se han rellenado. En caso de estar rellenados los 2 campos se validará que el email y password existen y si son correctos va directamente a la pantalla de plataformas y películas. En caso de no ser correcto se mostrará un aviso de tipo Toast informando que los datos introducidos no son correctos.
- Crear cuenta: Haciendo clic se mostrará otra actividad donde se mostrará un formulario con los campos necesarios para el alta de la cuenta.

**Pantalla de Registro**: Mostrará un formulario con los datos del usuario para realizar el registro de este. Las acciones a realizar son:

- Guardar: Realiza las validaciones necesarias, si es correcto guarda los datos y va directamente a la pantalla de plataformas y películas. Si no son correctas las validaciones muestra un mensaje Toast informando de los errores que se han producido.
- Cancelar: Muestra un mensaje de advertencia de tipo AlertDialog preguntando al usuario si está seguro de querer cancelar. En caso afirmativo se debe cerrar esta actividad y volver a la pantalla de login.

## COMPORTAMIENTOS 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 elemento, 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 (utilizar los tipos de datos que mejor se adapten a cada uno de los campos).

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

- Email (obligatorio)
- Nombre (obligatorio)
- Apellidos
- Fecha de nacimiento
- Tipo pregunta de seguridad, a elegir una entre (no obligatorio):
  - ¿Animal favorito?
  - ¿Ciudad favorita?
  - ¿Cuál es el nombre de tu mejor amigo?
- Respuesta pregunta seguridad
- Intereses: (a seleccionar uno o varios, al menos uno):
  - Tecnología

  - Deportes
  - Redes sociales
  - Cine
  - Otros (En caso de seleccionar este mostrar campo para escribirlo)
- Utilizar segundo factor de autenticación (Si / No)

**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:** Deberemos almacenar las diferentes películas. La información para guardar en SQLite será:

- 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 0 a 5 en función del propio usuario.

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 (serializado como un varbinary) 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.

## PARTE II

Se propone la realización de una amplicación de la práctica eligiendo alguno de los siguientes apartados:

- 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
    - DataStore
    - Room:https://developer.android.com/training/data-storage/room
- Realizar la internacionalización de la aplicación, por lo que cualquier literal debe estar traducido al menos a un idioma además del castellano (inglés por defecto).
- Se puede hacer cualquier otra ampliación adicional a la práctica previa consulta al profesor.

# 3. Evaluación

- **(8 puntos) PARTE I**
    - **(2 puntos) Pantallas de login, plataformas y**

**películas.**

- **(2 puntos) Almacenamiento de elementos en SQLite.**
- **(2 puntos) Almacenamiento de imágenes en BD o fichero físico.**
- **(1 punto) Exportación de listados en fichero en memoria externa.**
- **(1 punto) Calidad del código e interfaz de usuario.**

- **(2 puntos) PARTE II:**
  - **Utilización de técnicas avanzadas de visualización de interfaces gráficas.**
  - **Utilización de librerías de terceros.**
  - **Internacionalizar la aplicación.**
  - **Otros desarrollos adicionales.**

## Consideraciones a tener en cuenta

- Es obligatorio utilizar los ficheros de recursos para layouts, cadenas, imágenes, dimensiones, etc…
- Se deja total libertad para el diseño de las pantallas, se deben utilizar los controles y layouts acordes en cada caso.
- Se recomienda seguir las guías de diseño Material design.
- Hay que controlar todos los errores posibles (en caso de producirse se informará al usuario mediante un mensaje de tipo Toast por pantalla)
- Utilizar un código limpio, legible, documentado y comentado (siempre que sea de utilidad).
- Utilizar nombres descriptivos para los componentes de interfaz de usuario.
- Se recuerda que para obtener una buena disposición de los controles se deben ir anidando los layouts necesarios.
- Es necesaria la separación de las diferentes clases utilizando espacios de nombres.

## Otras recomendaciones

Es recomendable abordar la práctica en pequeños trozos agrupados por funcionalidades. Mi recomendación es:

- Empezar por el diseño de pantalla login y registro.
- Realizar la persistencia en sQLite de usuarios.
- Realizar las pantalla para plataformas.
- Realizar las pantallas para peliculas (será un clon de las pantallas de plataformas).
- Hacer exportación de datos a fichero.
- Ampliaciones.

## 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 Studio Electric Eel (última versión enero 2023)
    - Android 3.5.3 (máquina prácticas)
- Android SDK 13 API 33
- Minimum API Level: API 24: Android 7.0 (Nougat).

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

**Junto con la práctica hay que realizar una memoria explicativa (no más de una o dos hojas) con los pasos seguidos, la estratégia para abordar la práctica, problemas encontrados así como indicar en detalle lo que se ha abordado de la PARTE II (no hacer pantallazos de código).**

## Recursos

La documentación oficial proporcionada por Android es muy buena y dispone de muchas guías que pueden ser de utilidad para la realización de la práctica.

Página principal https://developer.android.com/

Guías de aspectos básicos https://developer.android.com/guide

Guías de interfaz de usuario https://developer.android.com/guide/topics/ui

Estilos

- https://developer.android.com/guide/topics/ui/look-and-feel/themes
- https://material.io/resources/icons/?style=baseline
- https://material.io/ (Material design)

Se recomienda encarecidamente que al alumno utilice estos enlaces de documentación oficial en lugar de estar buscando por enlaces cualquiera de internet.

# 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.