Trabajo de Aplicaciones Telematicas

Vidal Domínguez, Mario Chirlaque Hernández, Luis

Valencia Sellés, Ivan

June 21, 2020

Índice

1 Manual de usuario **3**
1.1 Cartelera 3
1.2 Búsqueda 4
1.3 Filmoteca 5
1.4 Pestaña de pelicula/serie 6

2 Memoria del trabajo **7**
2.1 Diseño 7
2.1.1 Disposición de los datos 7
2.1.2 Archivos xml 7
2.2 Scrapping 7
2.3 Internal Storage 8
2.3.1 ¿Cómo guardamos los datos? 8
2.4 Posibles bugs 10
2.5 Dificultades 10

1 Manual de usuario

Hemos querido realizar una app de películas de Filmaffinity. Su utilidad es recomendarnos nuevas películas, consultar cualquier filme y poder guardarlo en nuestra filmoteca personal para consultarlo con posteridad.

1.1 Cartelera

La aplicación al ser lanzada nos muestra una splash screen con el logo de la aplicación y nos deja en la **Cartelera**. Desde aquí tenemos tres paneles por los que podemos navegar para ver cuales son las nuevas películas que se ofertan.

Figure 1: Cartelera dentro de la App

1.2 Búsqueda

En la pestaña *Buscar* podemos examinar filmes gracias a el buscador de la parte superior. A continuación nos saldrán las resultados que más coincidan con nuestra búsqueda. Podremos entonces seleccionar cualquiera de ellos e inspeccionarlos en la pestaña de descripción.

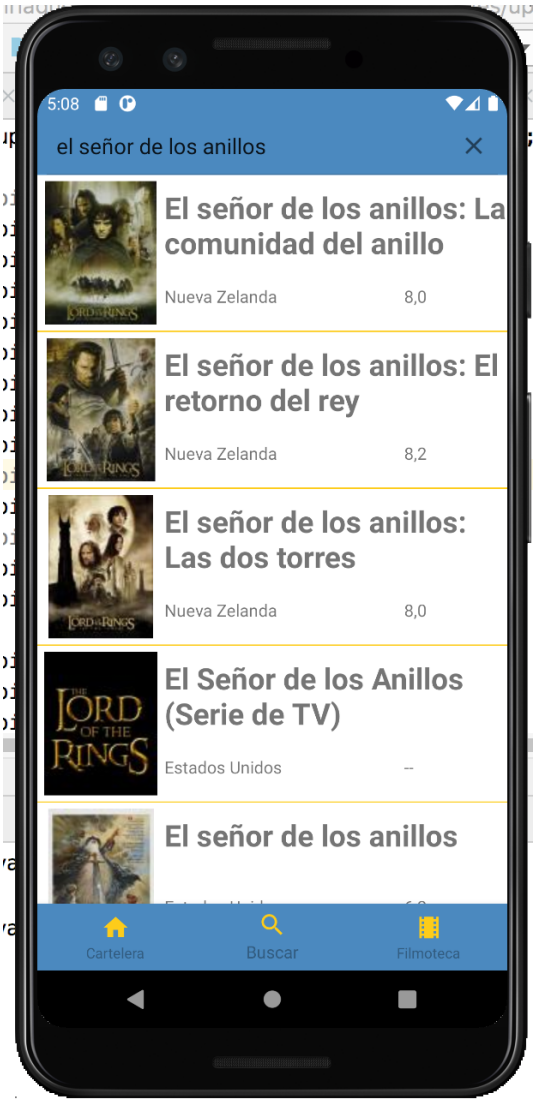


Figure 2: Búsqueda dentro de la App

1.3 Filmoteca

La tercera pestaña es la *Filmoteca*. En ella nos aparecerán todas las películas que hayamos seleccionado anteriormente. Así, podremos guardar las películas que nos interesen y poder revisarlas en cualquier momento, ya que se guarda en la memoria interna del dispositivo.

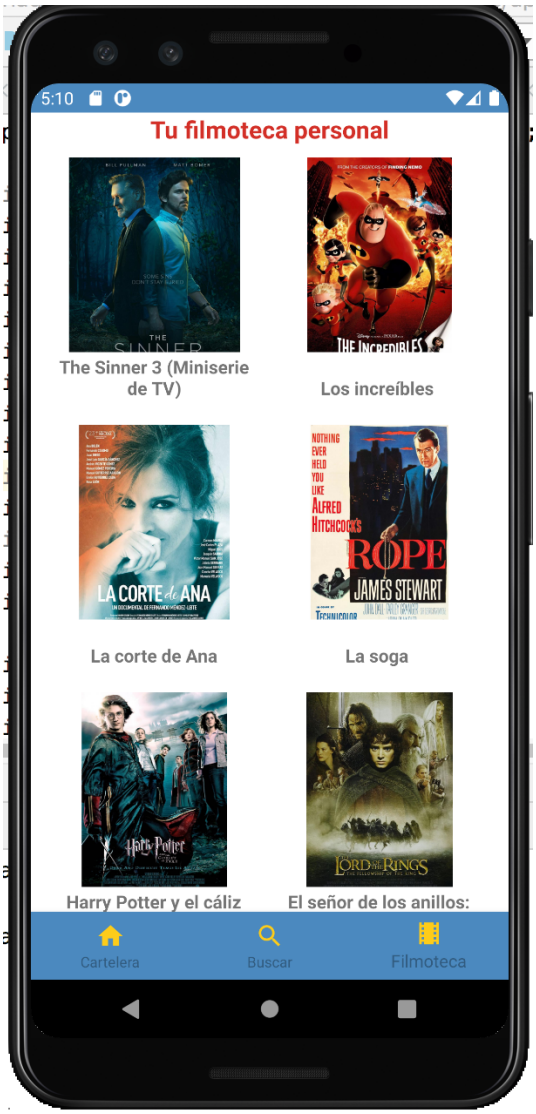


Figure 3: Filmoteca dentro de la App

1.4 Pestaña de pelicula/serie

En esta pestaña podremos ver la foto en grande del filme que hayamos seleccionado junto a otros datos de interés sobre ella. Adicionalmente nos aparecerá una casilla que podremos marcar en caso de desear guardar la película seleccionada en nuestra filmoteca particular, en caso de que queramos verla en cualquier otro momento.

Figure 4: Pestaña de una película dentro de la App

2 Memoria del trabajo

2.1 Diseño

Lo primero que realizamos en la aplicación fue organizar el diseño. Separamos la actividad principal en tres fragmentos: Cartelera, Buscar y Filmoteca. Además creamos una actividad extra (DescriptionActivity) que se abre cuando se pulsa cualquier película.

2.1.1 Disposición de los datos

Para la disposición de los películas por toda la aplicación hemos usado Recycler Views. En la cartelera hay tres Recycler Views Horizontales, en la búsqueda uno vertical y en la filmoteca uno vertical pero con dos columnas.
Para crear estas vistas hemos creado clases que heredan de RecyclerView. En estas clases se implementan los elementos visuales que luego son mostrados como los métodos onClickListener y onTouchListener para que el usuario pueda clickar e interaccionar de esta manera con los Recyler View. Al clickar se crea un *Intent* que pasa los datos de la película seleccionada a una nueva actividad que nos mostrará más información sobre ella.
Luego estos *Adapters* son instanciados en el fragmentos correspondiente. Es también en los fragmentos donde instanciamos las clases de datos en las que pasamos los parámetros de las películas para poder recabarlos luego en los adaptadores.

2.1.2 Archivos xml

Para cada fragmento hemos creado una disposición para mostrar los datos con un cierto orden para que fuera bonito e intuitivo de usar. Estos *xml* son **fragment_cartelera.xml**, **fragment_search.xml** y **fragment_filmoteca.xml**.
Además creamos también un archivo xml por cada tipo de *Recycler View* distinto que hemos querido implementar.
activity_main.xml contiene tan solo los fragmentos. Por último, **activity_description.xml** contiene la disposición de los datos de la película para que se puedan leer adecuadamente.

2.2 Scrapping

Para obtener los datos de Filmaffinity hemos utilizado Scraping Web, que consiste en extraer la información de los *HTML* mediante peticiones **HTTP**.

Para sacar la información de los html, hemos utilizado una librería llamada **jsoup**. Esta librería nos permite acceder a los html de las páginas web y navegar a través de ellos para encontrar la información que queramos. Dicha técnica ha sido empleada en nuestra app en sitios como la cartelera, la pestaña de pelicula/serie e incluso el buscador.

Por poner un ejemplo de uso de esta técnica voy a adjuntar parte del código del fichero:

Figure 5: Fichero DescriptionActivity.java

En este ejemplo lo que estamos haciendo es sacar datos que se encuentran dentro del HTML de la url que le proporciona el **AsyncTask**.

Lo primero que hacemos es conseguir el documento HTML y guardarlo en una variable de la clase *Document*, importada de Jsoup. Lo segundo que hacemos es coger todos los elementos que consigan nuestros criterios y los guardamos en una variable de la clase *Elements*, también importada de Jsoup. Por último lo que hacemos es recorrer todos los elementos que hemos seleccionado mediante la clase *Element* (importada de Jsoup) e ir cogiendo los que deseamos, en este ejemplo el título, el género, la fecha...

2.3 Internal Storage

Uno de los principales dilemas que tuvimos a la hora de hacer la aplicación era el de guardar los datos de nuestras peliculas. Había que decidir entre External Storage e Internal Storage, a pesar de que External Storage es más práctico y útil para casi cualquier uso, no disponíamos de los medios necesarios para emplearlo.

Al no disponer de un servidor para poder guardar los datos mediante External Storage acabamos deciendo utilizar Internal Storage que como su nombre indica guarda los datos internamente en el móvil. El incoveniente de este metodo de guardado es que a la hora de desinstalar la aplicación todos estos datos se pierden.

2.3.1 ¿Cómo guardamos los datos?

Los datos se guardan en un fichero json llamado **datos.json**, el cual su estructura es similar a esta.

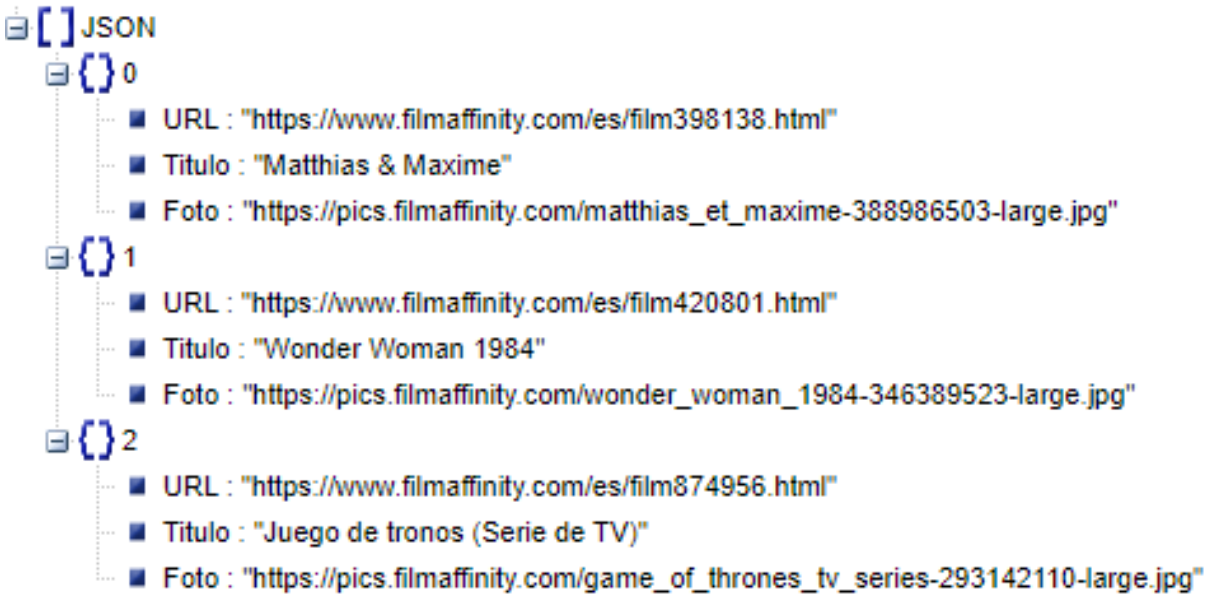


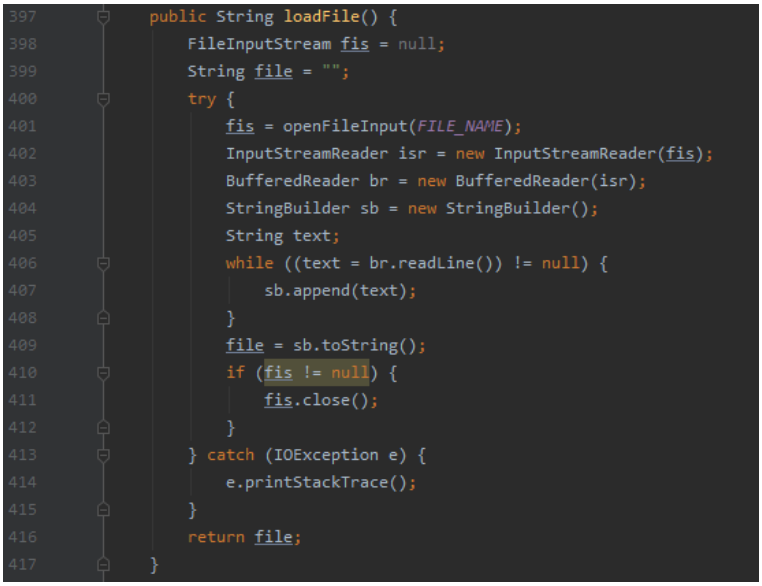
Figure 6: Fichero datos.json

En este fichero es donde guardamos lo necesario para que la pelicula o serie se pueda guardar en la filmoteca. El valor *URL* servirá para poder acceder después a conseguir más datos de dicha pelicula/serie. En cambio los valores *Título* y *Foto* simplemente servirán para poder mostrar en la filmoteca que película/serie es.

Mediante el uso de dos metodos **saveFile()** y **loadFile()** podremos manipular este fichero. A continuación muestro un ejemplo de estos ficheros:

```
381  @    public void saveFile(String datos) {
382           FileOutputStream fos = null;
383
384           try {
385               fos = openFileOutput(FILE_NAME, MODE_PRIVATE);
386               // URL
387               fos.write(datos.getBytes());
388               if (fos != null) {
389                   fos.close();
390               }
391           } catch (IOException e) {
392               e.printStackTrace();
393           }
394       }
395
```

Figure 7: Método saveFile() en el fichero DescriptionActivity.java



```
397     public String loadFile() {
398         FileInputStream fis = null;
399         String file = "";
400         try {
401             fis = openFileInput(FILE_NAME);
402             InputStreamReader isr = new InputStreamReader(fis);
403             BufferedReader br = new BufferedReader(isr);
404             StringBuilder sb = new StringBuilder();
405             String text;
406             while ((text = br.readLine()) != null) {
407                 sb.append(text);
408             }
409             file = sb.toString();
410             if (fis != null) {
411                 fis.close();
412             }
413         } catch (IOException e) {
414             e.printStackTrace();
415         }
416         return file;
417     }
```

Figure 8: Método loadFile() en el fichero DescriptionActivity.java

Para la manipulación de los ficheros Json hemos utilizado dos librerias distintas, la primera es la que ya trae por defecto Android Studio, **org.json**, y la segunda es la libería **javax.json**.

2.4 Posibles bugs

Debido a la técnica del scrapping no controlamos los archivos HTML, esto hace que si la página web de Filmaffinity cambia sus archivos HTML, posiblemente el código empleado necesite unas modificaciones antes de volver a funcionar correctamente.
También debido a la técnica del Scrapping es posible que el servidor de Filmaffinity detecte al usuario como un robot, haciendo que no cargue ni la cartelera ni ninguna busqueda. Esto solo sucedería si la aplicación se utilizase durante un periodo de tiempo relativamente pequeño, haciendo uso de muchas busquedas o refrescando la página de cartelera constantemente. La solución a este problema sería entrar desde un ordenador/móvil que este conectado a la misma red desde la que ha sucedido esto y hacer la prueba de *No soy un robot* que te pondrá la página web de Filmaffinity al entrar.

2.5 Dificultades

Las mayores dificultades que encontramos fue la descarga de las imágenes de la página web, ya que nos fue imposible efectuarlo con *ImageView* y tuvimos que hacerlo con *WebView*, cosa que resulta un poco mas costoso por su comportamiento.
La escritura y lectura de los datos en el almacenamiento interno también ha sido dificil de implementar ya que tenía que interactuar con los *CheckBox*, sacar los datos y escribirlos o leerlos, y luego eliminarlos en caso de quitar el marcador.