## 02. Utilizar ListAdapter como adapter del RecyclerView

Es la evolución de la aplicación anterior. En lugar de crear un adapter extendiendo directamente de la clase RecyclerView.Adapter, se utiliza una subclase de ésta facilitada por Android. El adaptador pasa a ser un ListAdapter.

El motivo es que ListAdapter mejora el rendimiento del RecyclerView. Cuando se le indica al anterior adaptador la lista con la información (List<Film>), se forzaba a volver a mostrar todos los ítems (invocando a notifyDataSetChanged() ). En ListAdapter no sucede eso. En su lugar hace una comparación de los ítems de la nueva lista (new) con los ítems de la lista vieja (old). Sólo mostrará (o eliminará de la pantalla) los ítems nuevos o que tengan valores distintos (o eliminará aquellos que hayan desaparecido en la nueva lista).

Para realizar esto, es necesario proporcionarle al ListAdapter un objeto DiffUtil.ItemCallback (en la aplicación el objeto estático DIFF definido en la clase Film). Este DiffUtil.ItemCallback exige definir dos métodos:

- **`areItemsTheSame()`**: Este método indica cuándo dos ítems (el de la lista old y el de la lista new) representan la misma información. Asumiendo que los ítems de la lista son entidades, dos ítems representan la misma información cuando las identidades de sendas entidades sean iguales. Recuérdese que la identidad de una entidad nunca cambia (el id del Film).
- **`areContentsTheSame()`**: Este método indica cuándo dos ítems que representan la misma entidad son iguales. Es decir, cuándo dos ítems contienen objetos iguales (la película new es igual a la película old y, por lo tanto, no ha sufrido ninguna modificación)

En ListAdapter ya está perfectamente definido el método **`getItemCount()`** de RecyclerView.Adaper y por consiguiente no hay que redefinirlo.

Para asignar los datos al adapter, ListAdapter incluye el método **`submitList().`**